Inxhinieria Softuerike

Konceptet në Testim të Softuerit

FAKULTETI: SHKENCA KOMPJUTERIKE DHE INXHINIERI

Kontenti

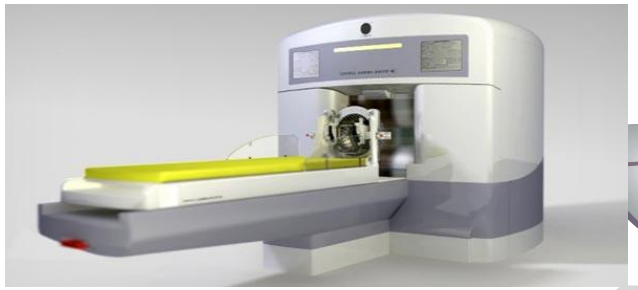
- Konceptet e testimt të softuerit
- Nivelet e testimit të softuerit
- Llojet e testimit të softuerit

konceptet e Testimit të Softuerit

PJESA E PARË

Softueri është gjithëandej











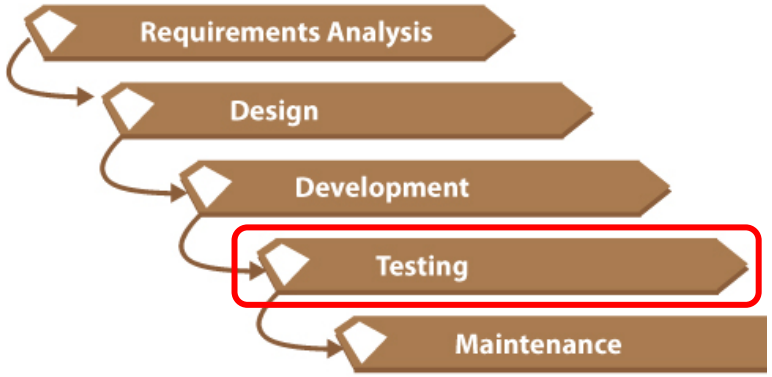


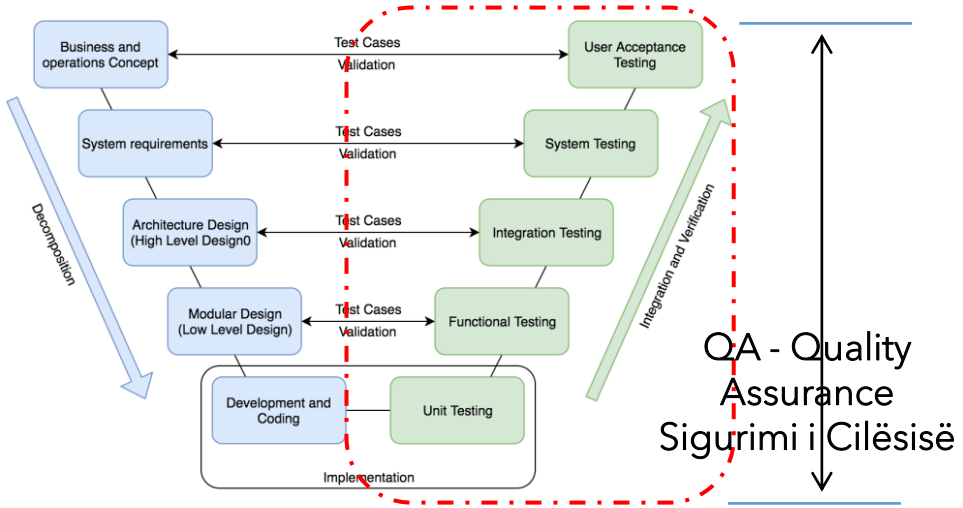

Dështime të Softuerit

- 2017: 606 dështime të regjistruara të programeve kompjuterike, duke prekur 3.7 miliardë njerëz, 314 kompani, humbje financiare 1.7 trilion dollarë
- 2016: Nissan tërhoqi 4 milion makina nga tregu për shkak të dështimit të softuerit në detektorët ndijor të kutisë së ajrit.
- 2016: Informacioni i humbur për shkak të butonit të mbrapa të shfletuesit gjatë përdorimit të softuerit në internet TurboTax
- 2015: Dështimet e terminalit tregtar të Bloomberg detyruan qeverinë britanike të shtyjë shitjen e borxhit prej 4,4 miliardë dollarësh
- 2014: Ndërprerja e Dropbox ishte për shkak të një faji në një skenar mirëmbajtjeje
- 2012: Fajet në një softuer të ri tregtar të Knight Capital shkaktojnë 440 milion dollarë
- 2003: Ndërprerja verilindore për shkak të sistemit të alarmit në dështimin e sistemit të menaxhimit të energjisë, duke prekur 40 milion njerëz në 8 shtete të SH.B.A.-së, 10 milion njerëz në Ontario, Kanada
- 1999: Toka Mars në NASA u rrëzua për shkak të një faji të integrimit në njësi

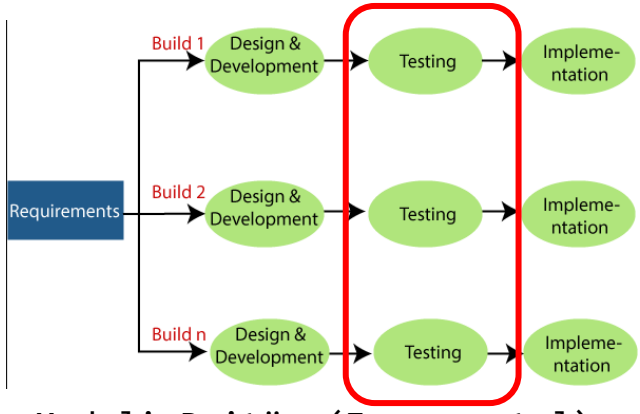
Kujtojm: Faza e testimi në modelt e ndryshem

Modeli Linear-Sekuencial



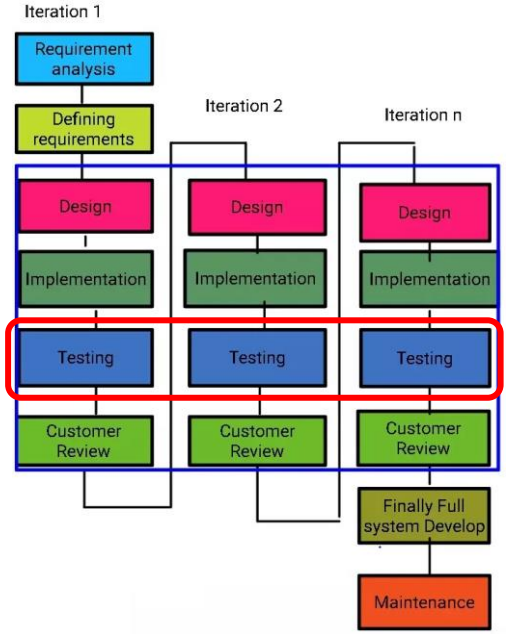


QA - Quality Assurance
Sigurimi i Cilësisë



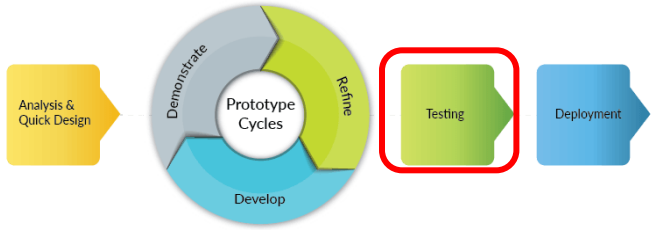
Modeli Rritës (Incremental)

Modeli Përsëritës (Iterative)

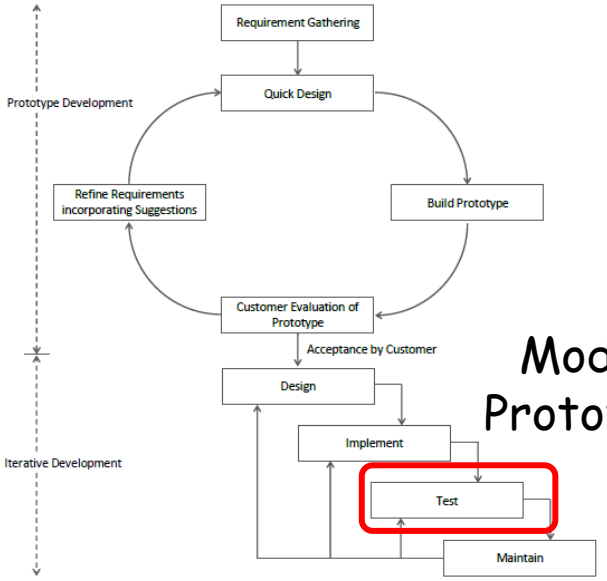


Kujtojm: Faza e testimi në modelt e ndryshem…

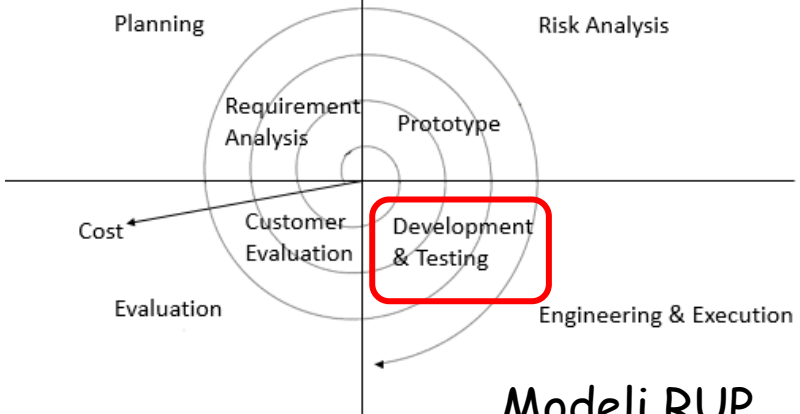
Modeli RAD



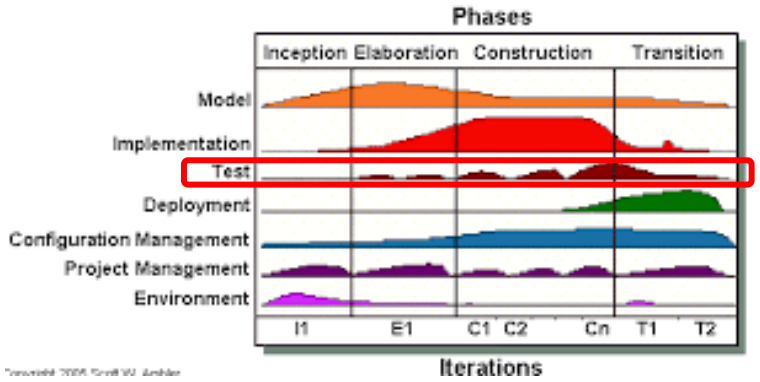
Modeli Prototipit



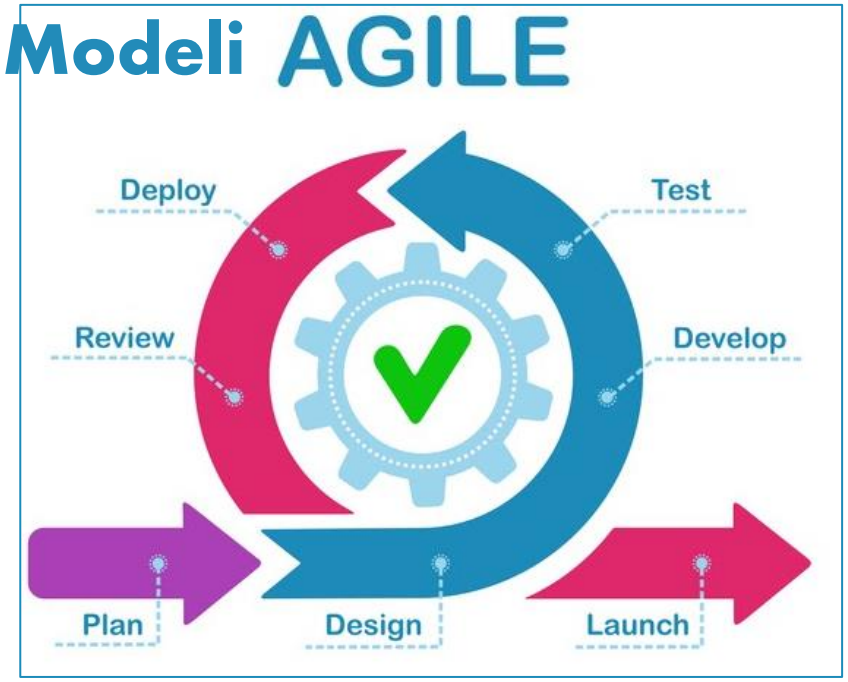
Modeli Spiral

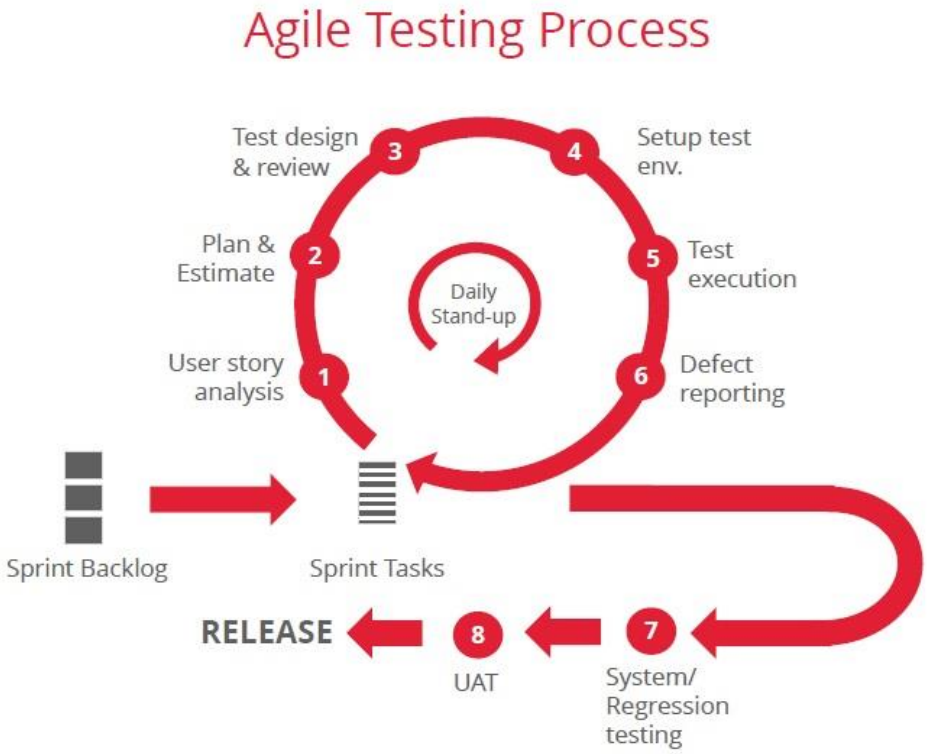


Modeli RUP



Kujtojm: Faza e testimi në modelt e ndryshem…





Pse duhët të Testojmë

water test

car test

spicy food

Buss crashes

Testimi në shekullin e 21-të

- ✓Softuer kritik i sigurisë, në kohë reale
- ✓Softueri i ngulitur (embedded)
- ✓Aplikacionet e ndërmarrjeve
- ✓ Aplikacion platforma për Siguri
- ✓Ueb aplikacione
- ✓Aplikacione mobile

Testimi i softuerit po bëhet më i rëndësishëm

Çfarë po përpiqemi të bëjmë kur testojm..?
Cilat janë qëllimet tuaja…?

Shembull i gabimeve të thjeshtë

```
public static int numZero (int[] arr)
{  // Efektet: Nëse arr is null throw NullPointerException
  // else return the number of occurrences of 0 in arr
    int count = 0;
    for (int i = 1; i < arr.length; i++)
       if (arr[i] == 0)
          count++;
    return count;
}
```

- Ekziston një gabim i thjeshtë në ***numZero***
- Ku është ***vendndodhja*** e ***gabimit*** në kodin burimor??
- Si do të rregullohet?
- A mund të ***arrihet*** vendndodhja e gabimit? Si sillet programi i ***korruptuar***? A e korrupton gjithmonë ***gjendjen*** e programit?
- Nëse gjendja e programit është e korruptuar, **dështon *numZero*? Si?**

Shembull i gabimeve të thjeshtë…

```
public static int numZero (int[] arr)
{   // Effects: If arr is null throw NullPointerException
  // else return the number of occurrences of 0 in arr
    int count = 0;
    for (int i = 1; i < arr.length; i++)
       if (arr[i] == 0)
          count++;
    return count;
}
```

- **Fault**: një defekt në kodin burimor

  `i = 1 [duhet të fillojnë kërkimin në 0, jo në 1]`

- **Error**: gjendje e gabimi në program është shkaktuar nga ekzekutimi i **defektit**

  `Bëhet 1 [hyrja në varg 0 nuk lexohet kurrë]`

- **Failure**: Përhapja e gjendjës së gabuar rezulton në dështim (failure) të programit

  (apli `Ndodh përderisa arr.length > 0 dhe arr[0] = 0`

Shembull i gabimeve të thjeshtë…

```
public static int numZero (int[] arr)
{   // Effects: If arr is null throw NullPointerException
  // else return the number of occurrences of 0 in arr
    int count = 0;
    for (int i = 1; i < arr.length; i++)
       if (arr[i] == 0)
          count++;
    return count;
}
```

Fault: `i = 1` [duhet të fillojnë kërkimin në 0, jo në 1]

- Test 1: [4, 6, 0], pritet 1

  Error: `i` është 1, jo 0, në iteracioni e parë
  Failure: asnjë

- Test 2: [0, 4, 6], pritet 1

  Error: `i` është 1, jo 0, gabimi përhapet në ndryshore ***count***
  Failure: **count** është 0 në gjendjen e kthimit

Motivi për të testuar softuerin:

*Si shpërndahet **kostoja** për Projktin?*

- Është interesante të **dimë cilat faza të një projekti** (zhvillimit të softuerit) kushtojnë më shumë para? Figura 1.1 paraqet shifra tipike.

- Është e qartë se *kostoja e testimit është e madhe*, ndërsa **kodimi përbën** vetëm *një pjesë të vogël* të **zhvillimit të softuerit.**

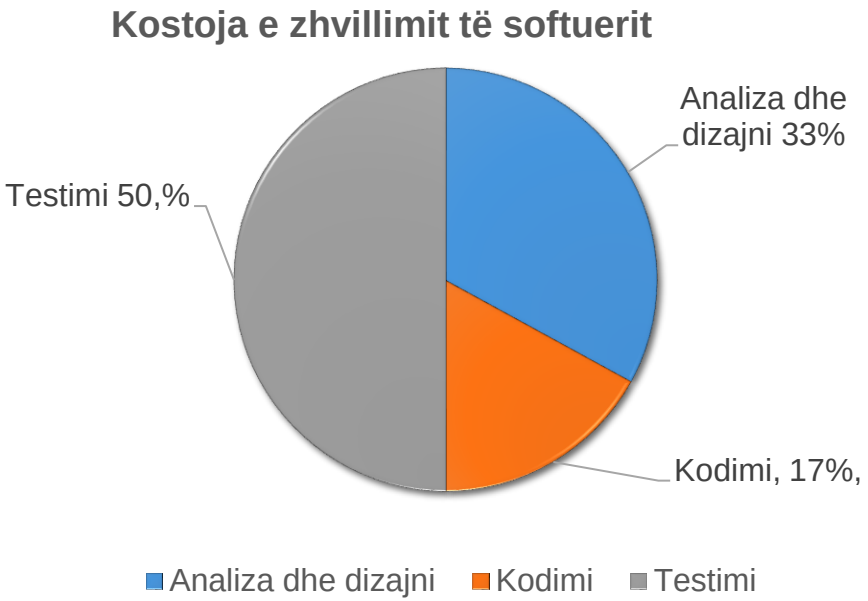


Fig. 1.1 Shpenzimet relative të fazave të zhvillimit të softuerit

Motivi për të testuar softuerin:
*ku ndodhin **gabimet** në Projkt?*

- Nëse gabimet janë një **problem i madh,** kur bëhen ato? Figura 1.2 tregon *shifrat duke* treguar numrin e gabimeve të bëra në fazat e ndryshme të një projekti:

- Megjithatë, **këto të dhëna** janë mjaft *relative*

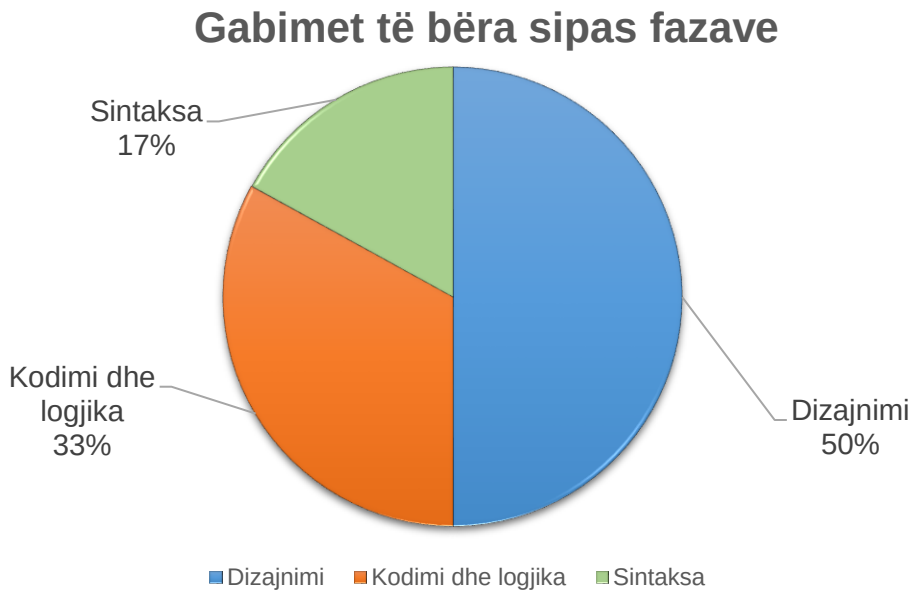
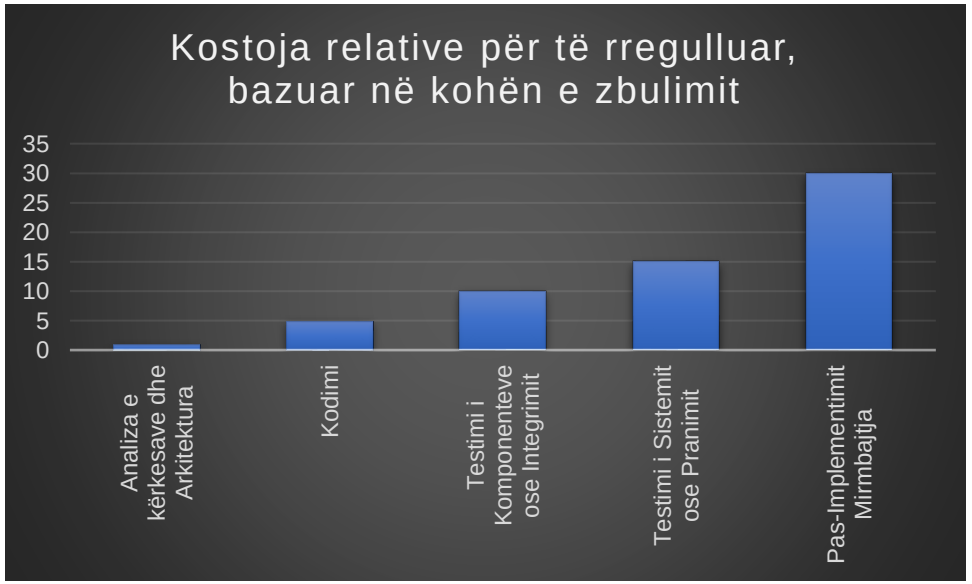


Fig. 1.2 Numri relativ i gabimeve të bëra gjatë fazave të zhvillimit të softuerit

Motivi për të testuar softuerin:
shpenzimet e rregullimit të softuerit?

- Ajo që ka rëndësi është se *sa kushton* për të **rregulluar një gabim**.
- Dhe sa më gjatë që gabimi të mbetet i *pazbuluar*, aq më *shumë shpenzimet* do të kemi për të i *rregulluar* këto *gabimet*.



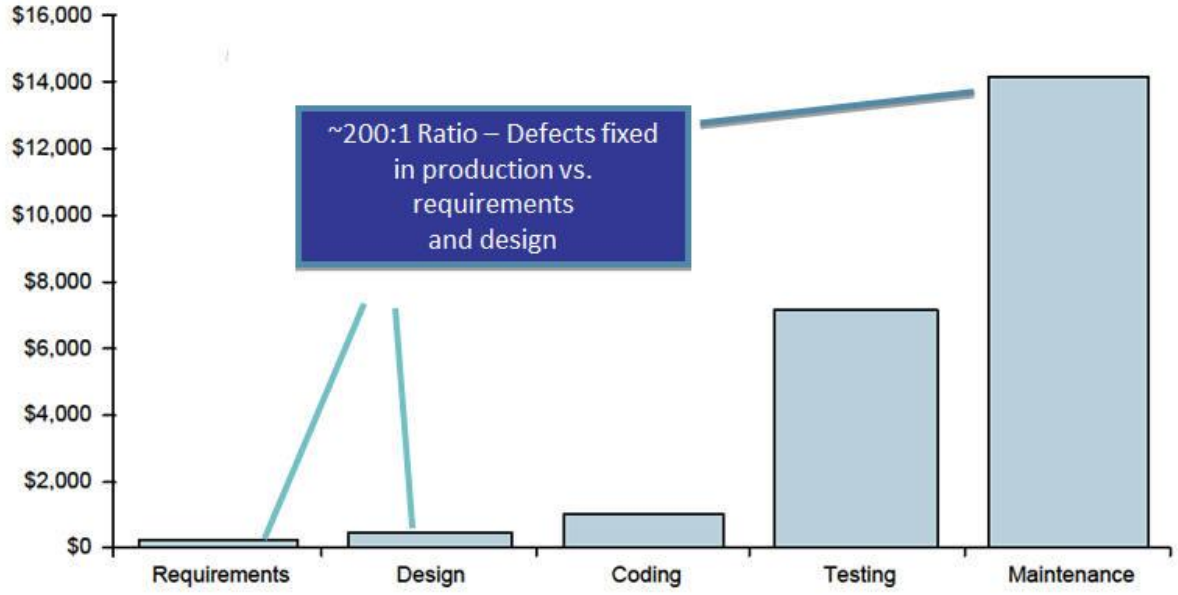
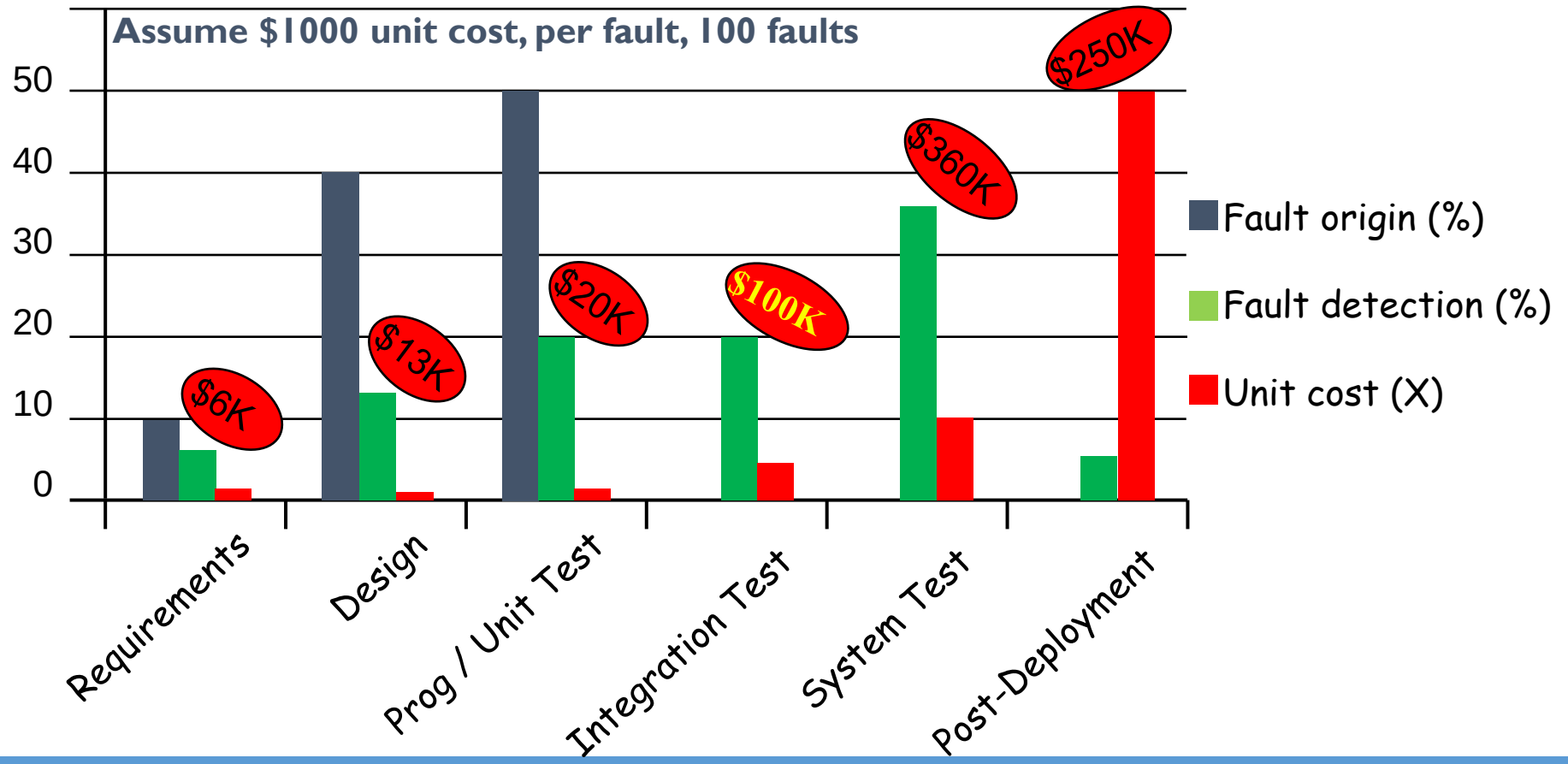


Fig. 1.3 source: *national institute of Standarts and Technology (USA*)

Kostoja e Testimit të Vonëshem



Motivi për të testuar softuerin: rasti në microsoft

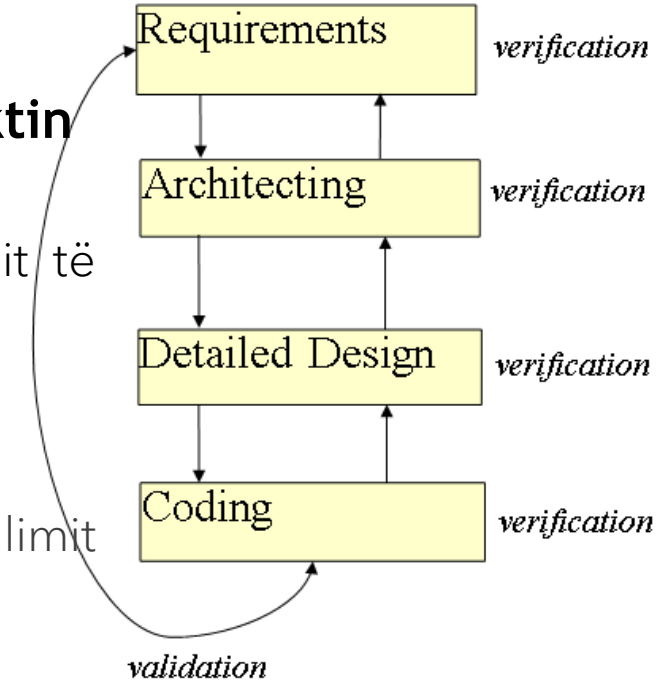
- **Testimi konsumon** në mënyrë tipike një proporcion të madh (*ndonjëherë deri në 50%*) të punës **për zhvillimin e një sistemi**.

- Microsoft-i punsonë *ekipet e programuesve* (të cilët programojn aplikacione) dhe **ekipet** plotësisht të *ndara të testuesve* (të cilët i testojnë ato).

- Në Microsoft ekzistojnë *sa ka puntorë që programojn* ka edhe po aqë **puntorë të përfshirë në testim**.

Validimi dhe Verefikimi (IEEE) V&V

- Testimi modern përqendrohet kryesisht në dy gjëra:
  - verifikimi dhe validimi (V&V)
- **Verifikimi** përpiqet t'i përgjigjet pyetjes: "**A e ndërtuam produktin siç duhet?"**
  - Procesi i derteminimit nëse produkti në fazen e caktuar të procesit të zhvillimit të softuerit përmbushin kërkesat e përcaktuara gjatë një faze.
- **Validimi** përpiqet t'i përgjigjet pyetjes : **"A kemi ndërtuar produktin e duhur?"**
  - Validimi (Vlefshmëria): Procesi i vlerësimit të softuerit *në fund* të zhvillimit të softuerit për të siguruar *pajtueshmërinë* me përdorimin e synuar



V & V qëndron për "verifikim dhe validim të pavarur"

Qëllimi i testimit të Softuerit

- **Qëllimet e Testimit të Softuerit:**
  - Të **demonstrojë** se softueri i plotëson kërkesat e **tij**.
  - Të identifikoj *situata* në të **cilat sjellja e softuerit** është e pasaktë ose e padëshirueshme.
- **Testimi i defekteve** ka të bëjë me eleminimin ***e gjendjëve*** të padëshiruara të sistemit të tilla si rrëzimet (crashes) e sistemit, ndërveprimet e padëshiruara me sistemet e tjera, llogaritjet e pasakta dhe korruptimin e të dhënave

Qëllimi i testimit të Softuerit …

- Testimi për **Validim**
  - Të ju demonstrojë zhvillueseve dhe klienteve të sistemit, që softueri i plotëson kërkesat e tijë.

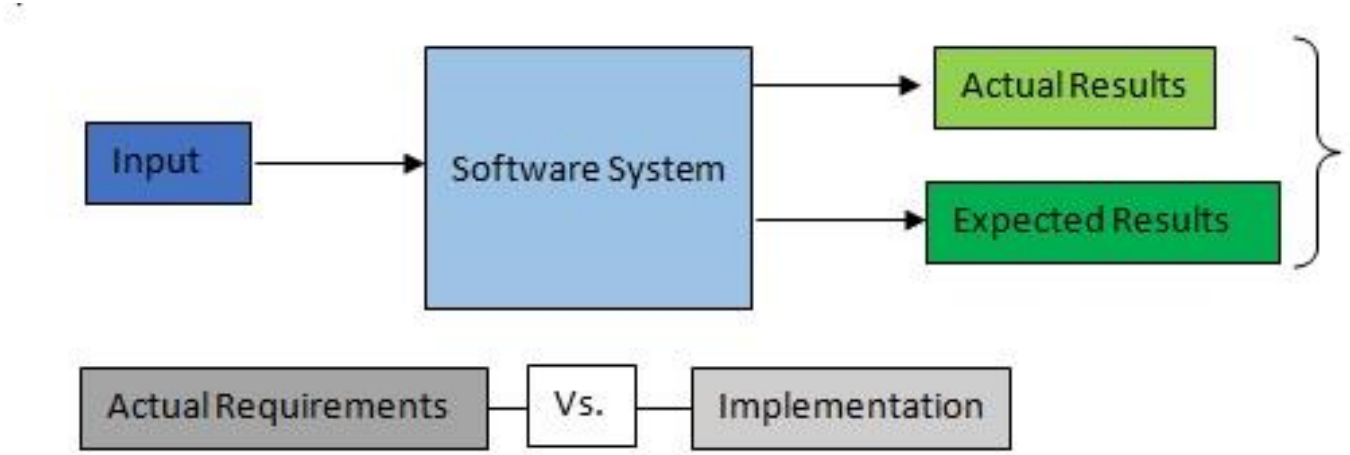
- Testimi për identifikim të **Defektit**
  - Për të zbuluar gabimet ose defekte në softuer ku sjellja e tij është e jo-korrekte ose nuk i përfille specifikat të definuar për të.
  - Një test i suksesshëm është një test që e shtynë sistemin të preformon në form jo-sakët..!! (gabimisht) dhe kështu ekspozon defektet në sistem.

Qëllimi i testimit të Softuerit…

- **Testimi** tregon nëse një program e bën atë që është menduar të bëjë dhe të identifikoj/gjejë defekte të programit para se të vihet në përdorim.
- **Në fazën e parë** - Testimi është procesi i ekzekutimit të një programi me qëllim të **gjetjes së gabimeve**.
- **Në fazën e fundit** - Testimi është procesi i demonstrimit, se gabimet nuk janë të pranishme (pas largimit të të gjitha bugs-ave)
  - Kur e testojm një program, duam të *shtojmë vlera* në të.
  - Shtimi i vlerave **përmes** testimit nënkupton ngritjen e cilësisë ose besueshmërisë së programit.
  - Rritjea e besueshmërisë së programit do të thotë gjetja dhe **eleminimin e gabimeve**.

Elementet bazë e procesit të Testimi të Softuerit

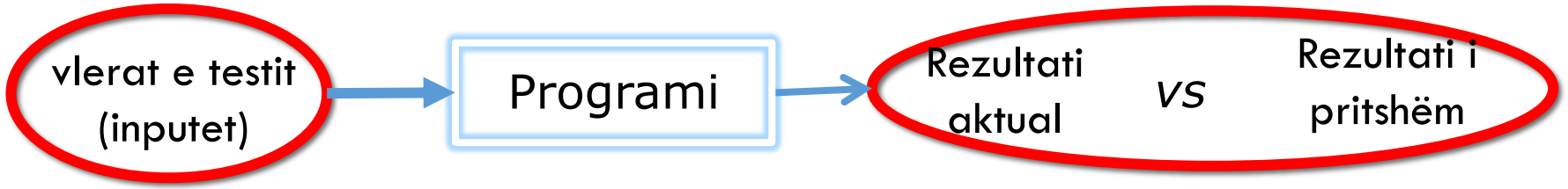
- **Testimi i softueri** është nje **aktivitet** që **kontrollon** nëse **rezultati aktual** përputhet me **rezultatin e pritur** dhe për të **siguruar** që dhe sistemi softuerikë është pa **defekte!!.**



Elementet bazë e procesit të Testimi të Softuerit...

- **Testimi** = proces i definimit së ***vlerave hyrëse*** (inpueve) për të *testuar* kundër një softueri
- **Debagimi** = proces i gjetjes së defekti ose defektëve, që ka rezultuar në dështimin e softuerit

`Test Rasti` përbëhet nga *`vlerat e testimit`* dhe ***`rezultatet e pritshme`***



1. Testimi është në thelb rreth zgjedhjes së grupeve të caktuar të vlerave nga domeni i inputeve të softuerit që testohet
2. Duke ofruar vlerat hyrse testues, krahasoni rezultatet aktuale me rezultatet e pritshme

Terme: Testimi, Test rasti dhe Plani i Testimit

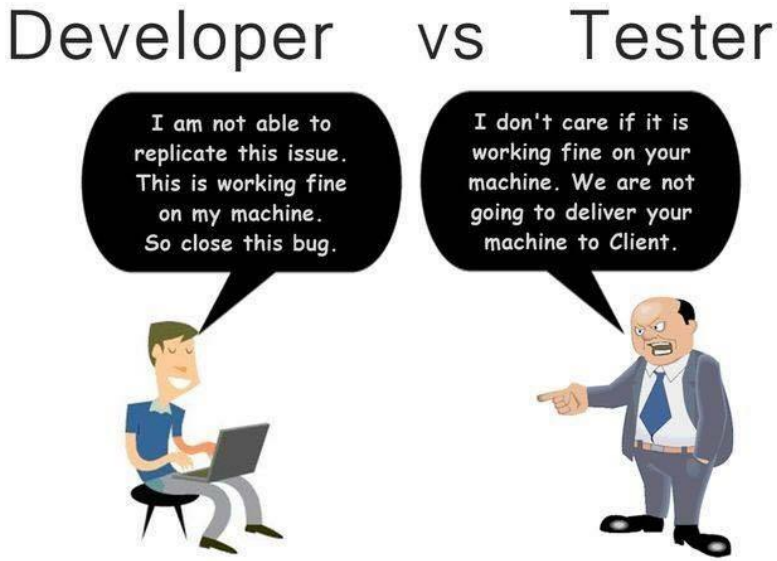
- **Termat e testim** dhe **testit të rast** përdoren në *mënyrë të ndërsjellë*. Në praktikë, të dyja janë të njëjta dhe trajtohen si **sinonime**.
- **Rasti i testimit (Test Case)** një grup i *inputeve për testim*, kushtet e ekzekutimit dhe rezultatet e pritshme të zhvilluara për një funksion të caktuar.
  - p.sh; Të testohoet një path i caktuar i programit ose të verifikojnë pajtueshmërinë me një kërkesë specifike.
- **Plani i Testimit** - është pjesë e dokumentacionit të përgjithshëm të projektit që përshkruan se çfarë, kur, si dhe kush do të përfshihet në procesin e testimit.

Terment e referencimit të elementeve të aplikacionit

- **Unit (Njësia):** testimi i njësive më të vogla mund të referencohemi (p.sh., funksioni, procedura, moduli, klasa e objektit, etj.)
- **Komponenti:** testimi i një koleksioni të njësive që përbëjnë një komponent (p.sh., programi, paketa, detyra, klasat e objekteve ndërvepruese, etj.)
- **Produkti:** testimi i një koleksioni të përbërësve që përbëjnë një produkt (p.sh., nënsistemi, aplikacioni, etj.)
- **Sistemi:** testimi i një koleksioni të produkteve që përbëjnë një sistem të dorëzueshëm.

Kush duhet të bëjë Testimin?

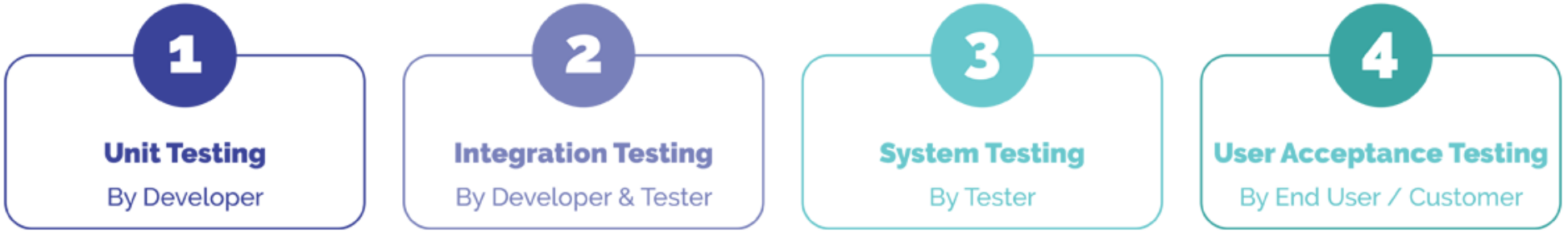
- **Testimi** kërkon që **zhvilluesit** të *gjejnë gabime* nga **softueri** i tyre.
- Është e *vështirë* për **zhvilluesi** i softuerit të *tregojë gabime* nga krijimet (programet) e veta.
- **Shumë organizata** kanë bërë një dallim në mes të *fazës së zhvillimit* dhe *testimit*, duke i bërë **grupe/departamente** të ndryshëm përgjegjës për secilën **faze/pjesë**.



Nivelet e bazë te testimi të softuerit

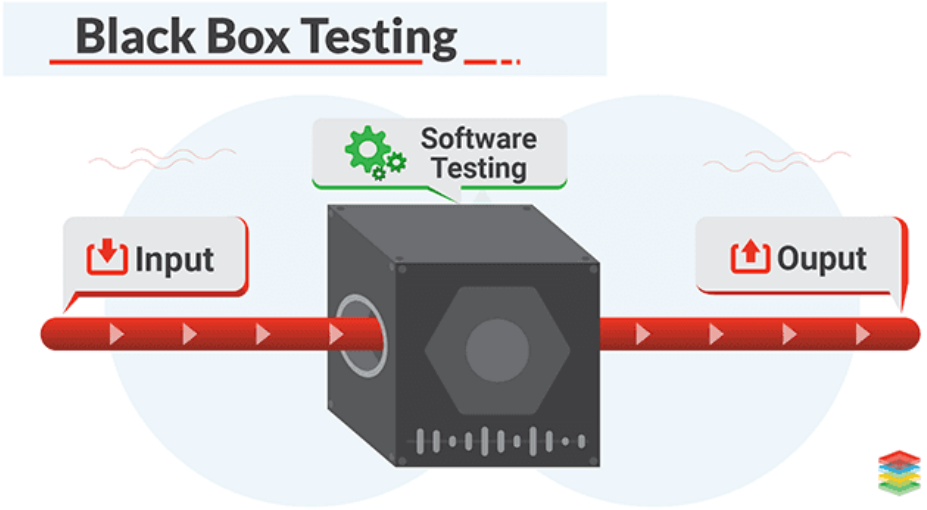
- Bazuar në komponenten e ndertimi të aplikacionit, ekziston nivelet e testimit të softuerit:
  - Testimi i **Njësis** (Unit Testing)
  - Testimi i **Integrimit** (Integration Testing)
  - Testimi i **Sistemit** (System Testing)
  - Testimi i **Pranimit** (Acceptence Testing)

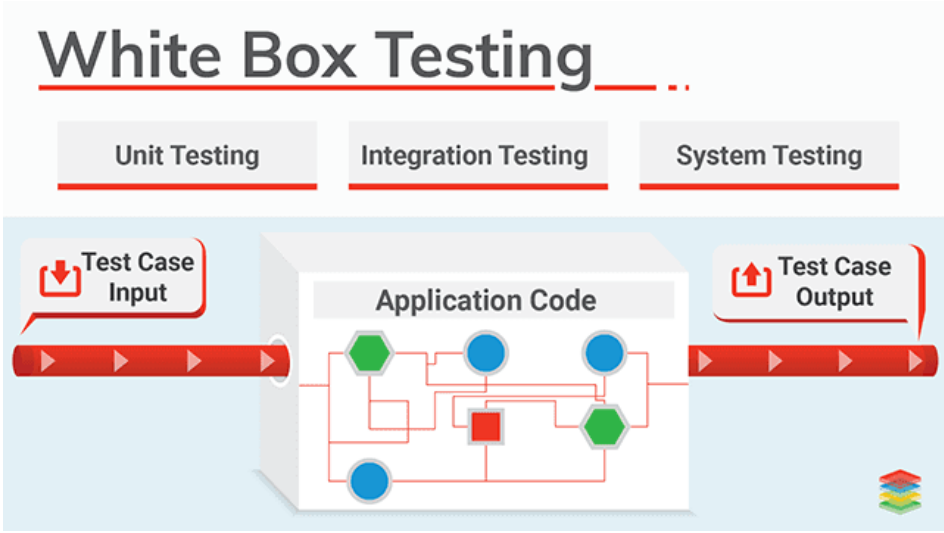
nivelet e Testimi të Softuerit



Teknikat per ralizimin e testimit

- **Black-box testing and White-box testing**
  - **Black-box testing** > Behavioral testing.
  - **White-box testing** > Structural testing.





Teknikat per ralizimin e testimit…

- **Çfarë është Testimi i Kutisë së Zezë (Black Box)?**
- **Testimi i Kutisë së Zezë referohet si Testim i Sjelljes (**Behavioral**).**
  - Është një metodë e testimit të softuerit ku struktura / dizajni / implementimi i brendshëm i elementi të testuar **nuk është i njohur për testuesin**.
  - Është një metodë e cila përpiqet të gjejë një gabim në kategoritë e mëposhtme:
    - Funksione të pasakta ose që mungojnë.
    - Gabimet e interfases/ndërfaqes.
    - Gabime në integrimin me një strukturë të dhënash ose qasjes të dhënash të jashtme.
    - Gabimet e sjelljes ose performances
    - .Gabimet e inicializimit dhe përfundimit.

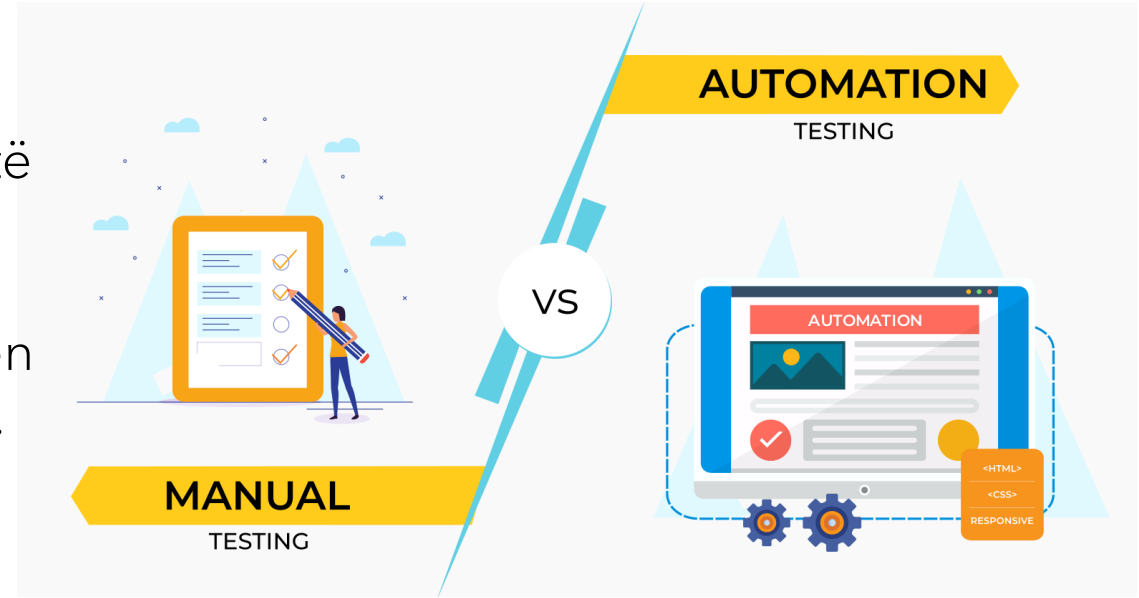
Teknikat per ralizimin e testimit…

- **Çfarë është Testimi i Kuti të Bardhë (white box testing)?**
  - Testimi i Kutisë së Bardhë është një lloj teknike testimi që kryesisht shqyrton strukturën e programit dhe nxjerr të dhëna të testit në bazë të logjikës ose kodit të programit.
  - Ai u referohej gjithashtu emrave si testimi i clear box testing, open box testing, logic-driven testing dhe path driven testing ose structural testing.
- Si funksionon Testimi i Kuti të Bardhë? Hapat për të kryer këtë Testim përmenden si më poshtë në një renditje specifike -
  - Së pari, të gjitha funksionet, komponentet dhe programet që do të testohen, identifikohen së pari.
  - Krijoni një grafik rrjedhës dhe identifikoni / vizatoni të gjitha shtigjet e mundshme në grafikun e rrjedhës.
  - Identifikimi i të gjitha shtigjeve të mundshme nga grafiku i rrjedhës.
  - Shkruaj test rastet (test cases) për secilën rrugë të vetme të shtegut të rrjedhës.
  - Ekzekutoni dhe përsëritni test rastet.

Testim **Manual** vs **Automatizuar**

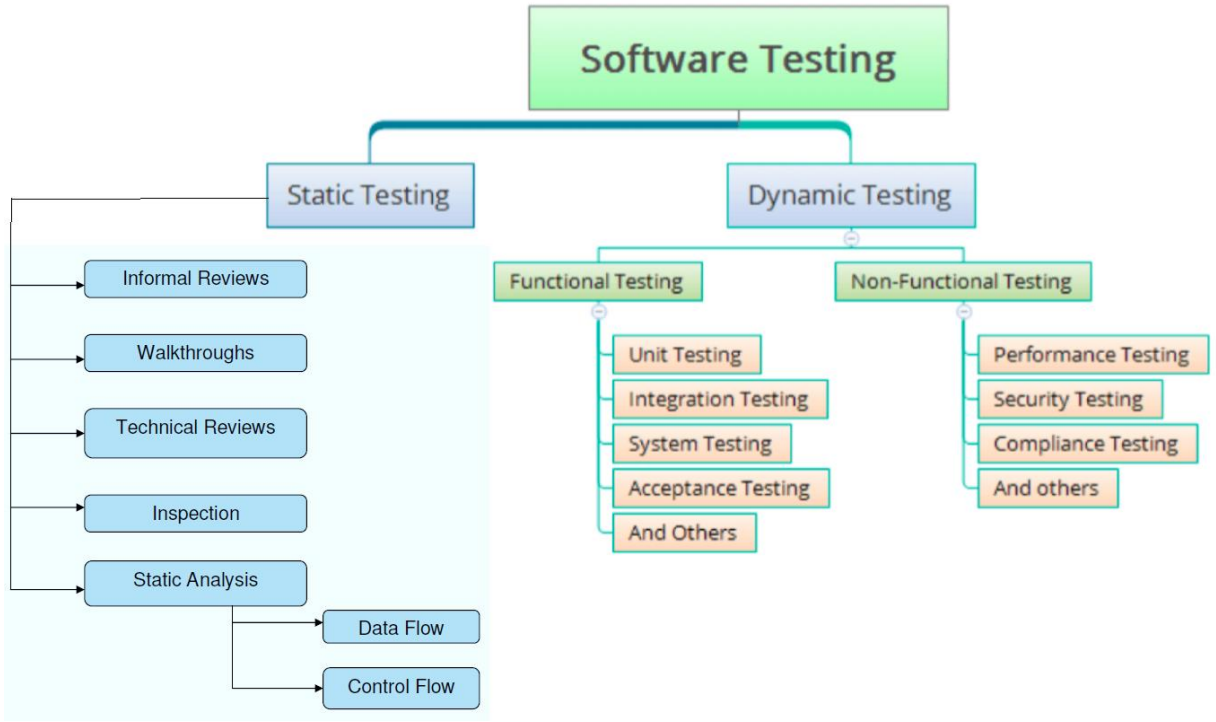
- Testim **Manual** vs Testim **Automatizuar**
- Testimi mund të bëhet **manualisht** ose duke përdorur një mjet të **automatizuar** të testimit:

  - **Manual** - Ky test kryhet pa marrë ndihmën e mjeteve të automatizuar të testimit.
  - **Automated** - Ky test është një procedurë testimi e bërë me ndihmën e mjeteve të automatizuar të testimit.



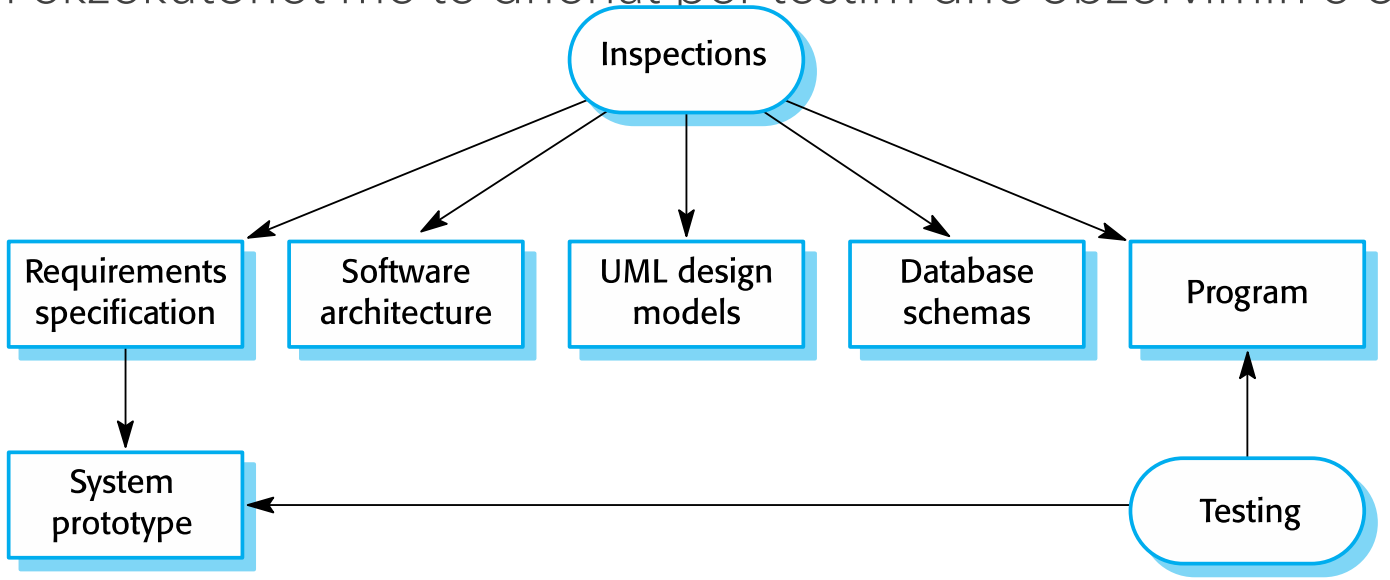
Testimi Statik dhe Dinamik

- **Testimi Statik** : Testimi programi i pa-ekzekutuar:
  - Inspektimi i softuerit dhe disa forma të analizës
  - Efektive në gjetjen e llojeve të caktuara të problemeve të tilla si problemet që mund të çojnë në gabime kur programi modifikohet
- **Testimi Dinamik**: Testimi duke ekzekutuar programin me ***vlera hyrse (inpute) reale***



Inspektimet dhe Testimet (*testimi static*)

- `Inspektimet softuerit-` Ka të bëjë me analizën e *aspektit statike të sistemit* për të identifikuar **problemet** (verifikim statikë ose testimi statik).
- `Testimi i Softuerit` - Ka të bëje me *ekzekutimin dhe obzervimin* e **sjelljeve** të produktit/programit (verifikimi dinamik ose testimi dinamik)
  - Sistemi ekzekutohet me të dhënat për testim dhe obzervimin e operimit të sjelljeve



Testimi në zhvillim(Testimi Dinamik)

- Testimi në zhvillim përfshin të gjitha **aktivitetet testuese** që kryhen parimisht nga **ekipi i zhvillimit** të sistemit.
  - *Testimi i njësisë (Unit Testing),* ku testohen njësitë individuale të **programit** ose klasat e objekteve. Testimi i njësisë duhet të fokusohet në testimin e *funksionalitetit* të objekteve ose metodave.
  - *Testimi i komponentëve (Component or Integration Testing),* ku janë **integruar** disa *njësi individuale* për *të krijuar komponente*. Testimi i komponentëve duhet të fokusohet në testimin e *interfaceve të komponentëve*.
  - *Testimi i sistemit (System Testing),* ku disa ose të gjithë komponentët në një sistem janë të **integruar** dhe sistemi testohet në tërësi. **Testimi i sistemit** duhet të fokusohet në *testimin e ndërveprimeve të komponentëve*.

Shembull: **Testimit Static** dhe **Testimit Dinamik**

- Rasti: ***Shporta e blerjeve Online***
- **Teknikat e testit statik:**
  - Rishikoni dokumentet e kërkesë, dokumentet e dizajnit konceptual
  - Kontrollimi i GUI të aplikacionit
  - Kontrollimi i strukturës së të bazës së të dhënave të aplikacionit.
- **Teknika të Testimit Dinamik:**
  - Testimi i funksionalitetit të faqes së ndryshme.
  - Kontrollimi i procesit të checkout dhe mënyrat e pagesës.
  - Testimi i ndërfaqeve midis faqeve të ndryshme.

Testimi i njësisë (Unit Testing)

- Teston çdo modul individualisht.
- Zbaton një testimi i teknikës së kutisë së bardh (logjikën e programit).
- E implementuar ose ralizuar nga zhvilluesit.

Testimi i komponentëve (Component ore Integration testing)

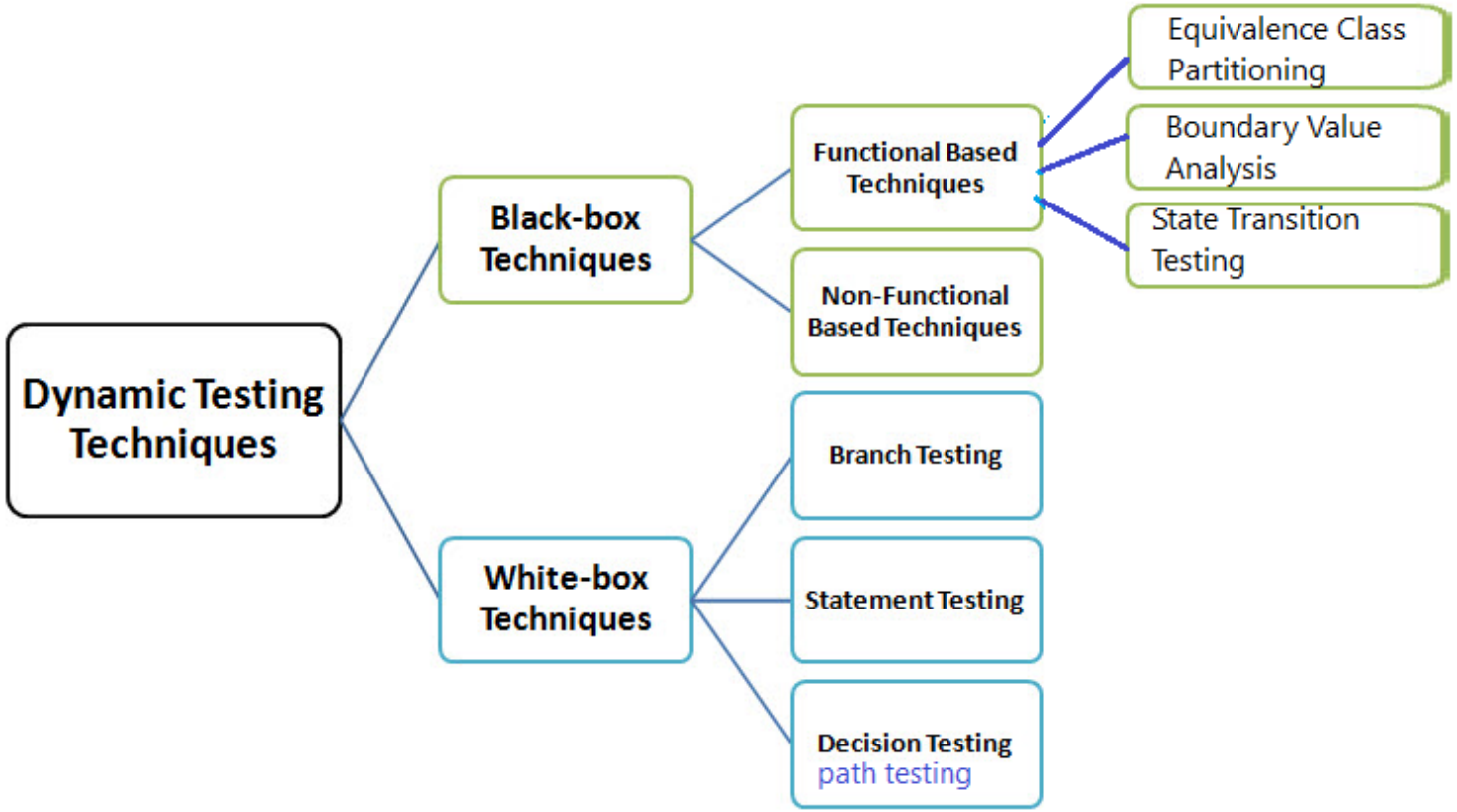
- Sapo të gjitha modulet të jenë testuar sipas *tëstimit të njësia*, pastaj kryhet testimi i integrimit/komponenteve.
  - Është testimi **sistematik**.
- Prodhoni teste për të identifikuar gabimet që lidhen me nderfaqet (interfacing).
- **Llojet:**
  - Testimi i Integrimit; Big-Bang
  - Testimi i Integrimit: Lartë-poshtë
  - Testimi i integrimit; Poshtë-lartë
  - Testimi i integruar: mix (hybrid)

Testimi i sistemit (Sistem testing)

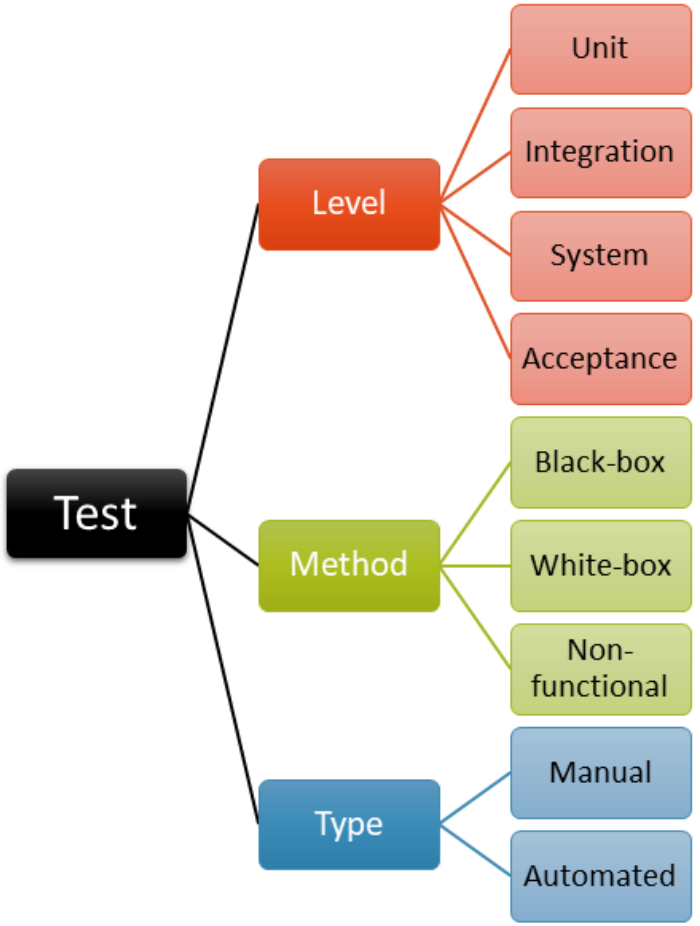
- **Sistemi në tërësi** është testuar për të *zbuluar/identifikuar gabimet e kërkesave*.
- **Verifikon** që të gjitha elementet e sistemit **funksionojnë** si duhet dhe se funksionimi dhe performanca e përgjithshme e **sistemit** është **arritur**.
  - Zbaton një testimi të teknikës së kutisë së **Zezë** (testimi funksional).

- Llojet testimit të sistemit:
  - Testimi i Alpha
  - Testimi i Beta
  - Testimi i Pranimit (acceptance)
  - Testimi i Performancës

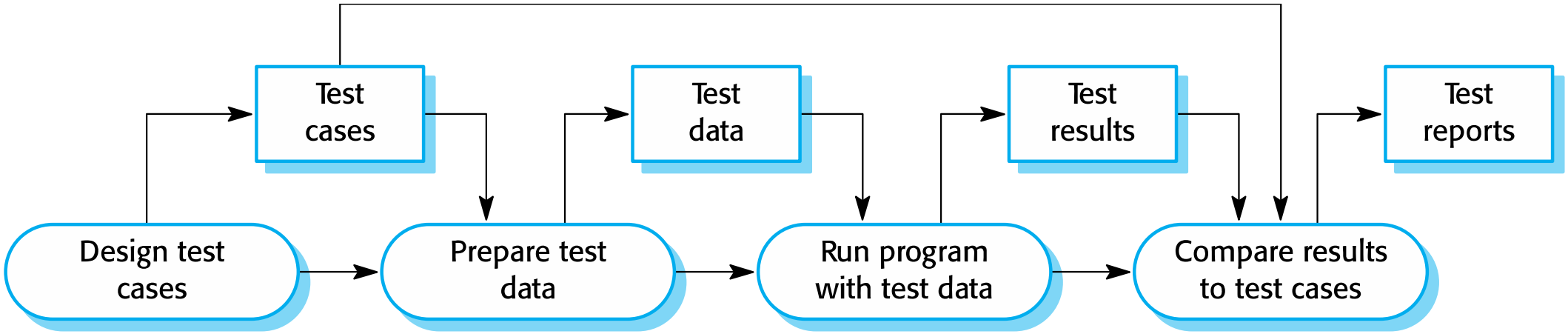
Teknikat/metodat e testimit Dinamik



Përmbledhje e Niveleve, Metotave dhe Llojeve të Testimi të Softuerit



Një model i procesit të Testimit të Softuerit



Terme: Mostra/Templata e rastit të Testimit

	Emri i Dritares Screen Name	Kalkulimi i Kredis	Logo e Kompanis
	Përgatitur nga Prepared By	Ramiz HOXHA	
	Designation	Test Analyst	
	Data		

ID e Skenarit	3.1	Përshkrim i Skenarit Scenario Description	Verifikoni funksionalitetin e funksionit Kalkulimit te Kredis					
3.No	ID Rastit Testues Test Case ID	Përshkrim Rastit Testues Test Case Description	Hapat e Testimit Test Steps	Vlerat hyrëse Input Values	Rezultati e Pritshme Expected Result	Rezultati Aktual Actual Result	Statusi Status	ID e Defekti Defect ID
1	3.1.1	Testimi për të kontrolluar funksionalitetin e funksionit të Kalkulimit te Kredis duke futur e të dhënave të vlefshme në fusha të obligueshme	Shto: Emrin Shto: Acc No Shto :Loan Shto :Term	Name: John Smith Acc No: 123456 Loan: 2500 Term: 3 years	Term: 3 years Repayment: 79.86 Interest rate: 10% Total paid: 2874.96	Term: 3 years Repayment: 79.86 Interest rate: 10% Total paid: 2874.96	Kaloi/Pass	
1	3.1.2	Testimi për të kontrolluar funksionalitetin e funksionit të Kalkulimit te Kredis duke futur e të dhënave të pa vlefshme në fusha të obligueshme	Shto: Emrin Shto: Acc No Shto :Loan Shto :Term	Name: J Acc No: 023456 Loan: 2500 Term: 3 years	Duhet të shfaqet në error mesazh 'Ju lutem shtoni te dhenat e rregulla'	Term: 3 years Repayment: 79.86 Interest rate: 10% Total paid: 2874.96	Nuk Kalon/Fail	

Përmbledhje: Pse e testojmë softuerin ?

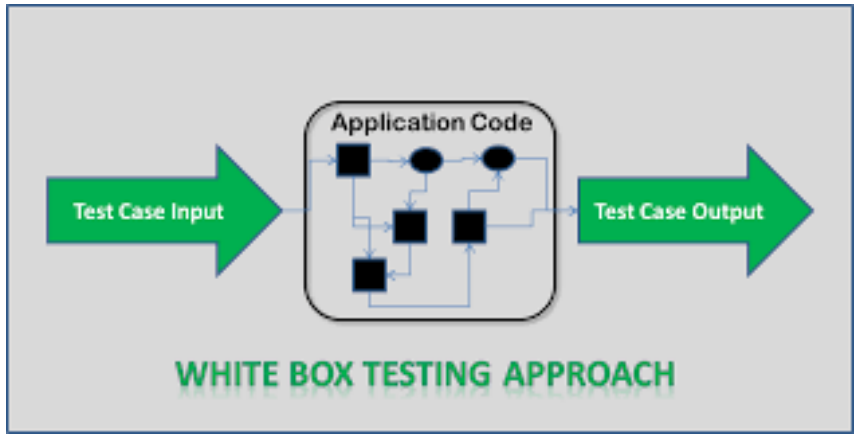
- Qëllimi i një testuesi është të eliminojë defektet sa më shpejt të jetë e mundur.
- Përmirësoni cilësinë
- Ulja e kostos
- Ruani kënaqësinë e klientit

Metodat e Testimit të Softuerit (sipas kutisë së zezë dhe kutisë së bardhë)

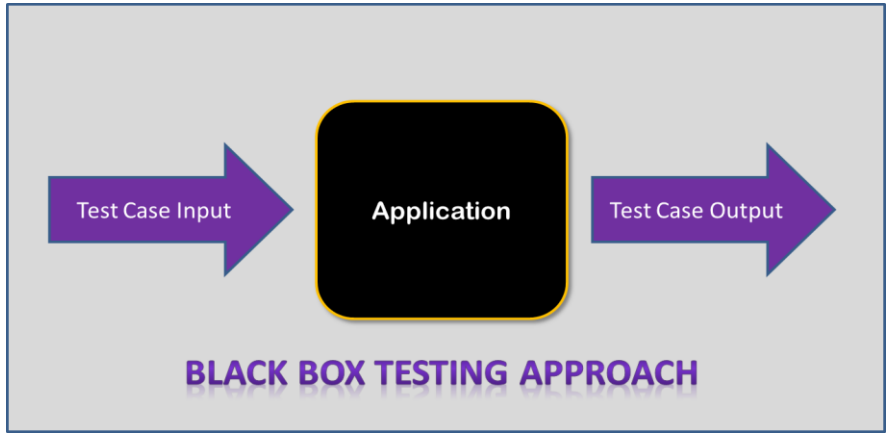
PJESA DYTË

Metodat e Testimit të Softuerit

- Të ***dy teknika*** themelore të testimit software, testimi sipas ***kutis së zezë*** dhe testimi sipas ***kutis së bardhë***



Metoda e testimit sipas kutis të bardhë

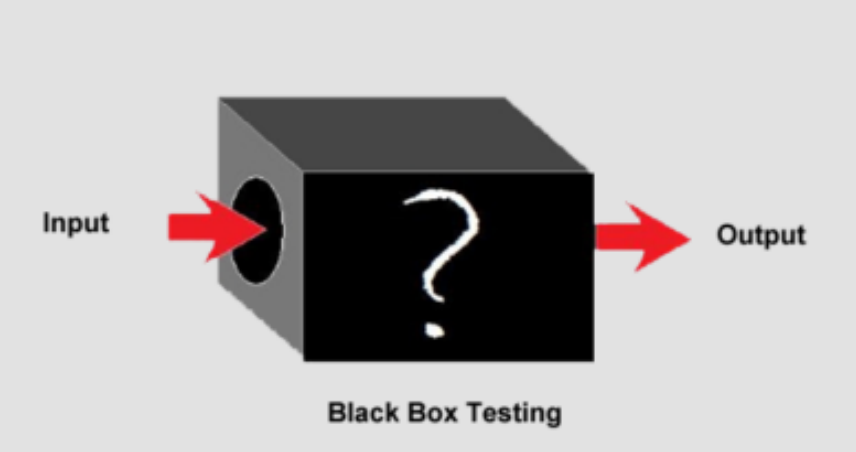


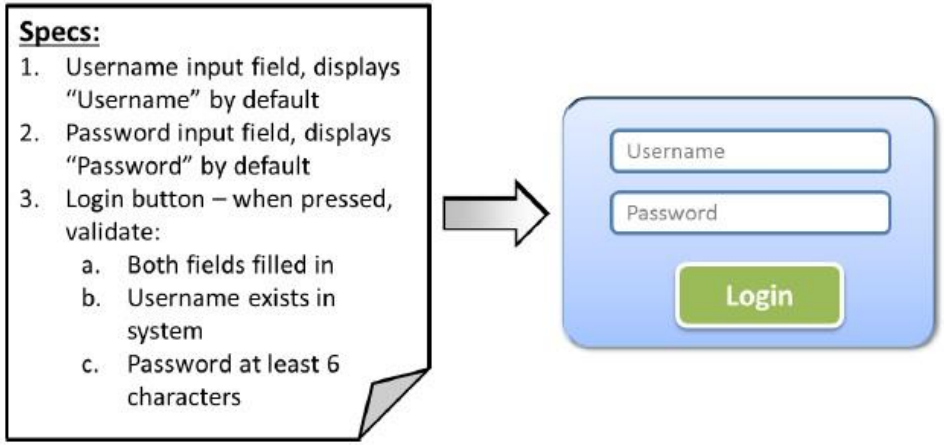
Metoda e testimit sipas kutis të zezë

Testimi i sipas metodës së kutis së Zezë

Testimi i sipas metodës **kutia e Zezë** (Black-box Testing) (1)

- Testimi i kutisë së zezë. Testimi i softuerit bazuar në kërkesat funksionale dhe të biznesit në funksionimin e tij pa njohuri të strukturës së brendshme ose kodit burimor të programit.
- Për shkak se **testimi** bëhet sipas kutis e zezë qëllimisht injoron strukturën e kontrollit të programit, vëmendja është përqendruar kryesisht në sferën e informacionit (p.sh., të **dhënat** hyrse- inputs dhe në të **dhënat** që dalin-

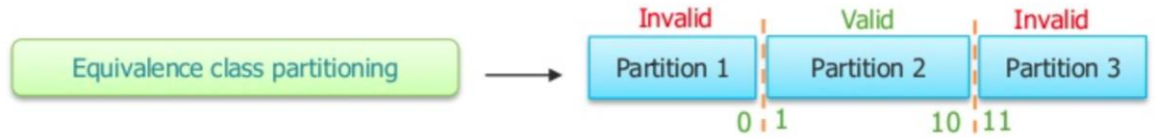


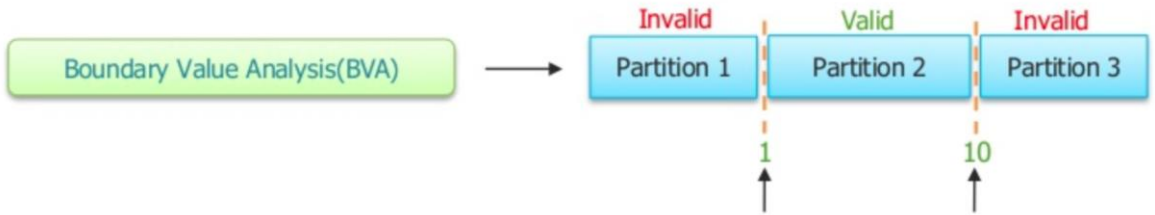


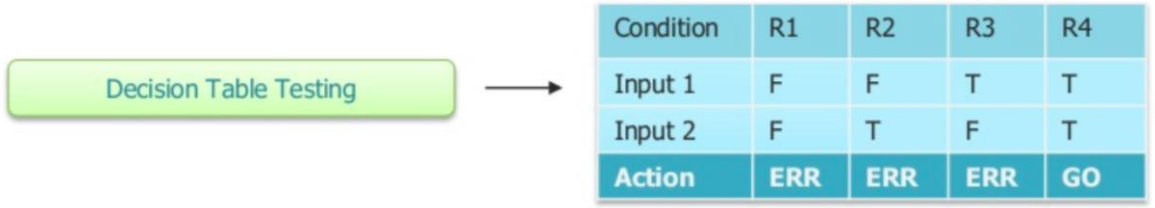
Testimi i sipas metodës **kutia e Zezë** (Black-box Testing) (2)

- Ekzistojnë **katër** teknika të *bazuara në specifikacione* ose teknika sipas ***kutisë së zezë:***
  - Ndarjes së Klasës ekuivalente (Equivalence Class Partitioning)
  - Analiza e Vlerës së kufirit (Boundary Value Analysis)
  - Testimi i gjendjës/kalimit të Tranzacionit (State Transition Testing)
  -

} Fokusuar në logjikën e biznesit ose rregullat e biznesit





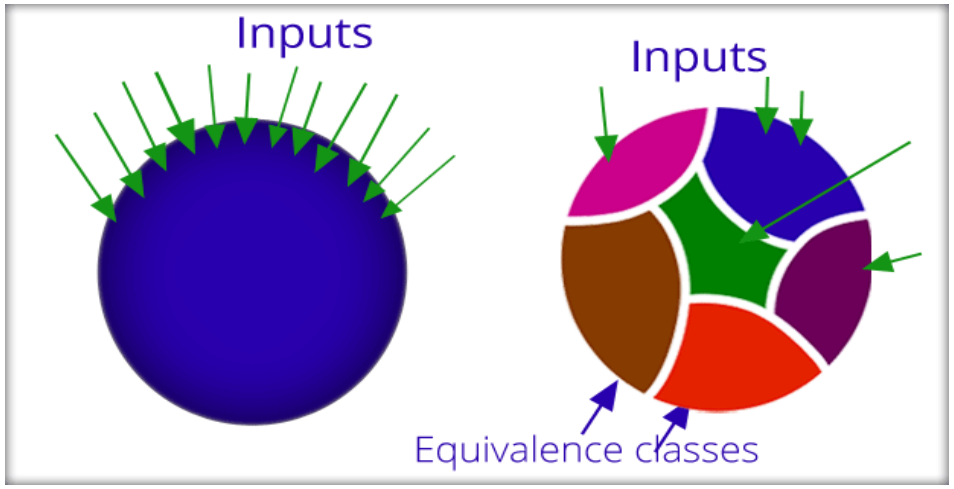


- Zgjidhja e mundshme do të ishte duke ndarë në 3 klasë të ndryshme:
  - Klasa e vlerave pozitive
  - Klasa e vlerave negative
  - Klasa e vlerave të pavlefshme/invalide

Testimi i sipas metodës **kutia e Zezë** (Black-box Testing) (3)

❑Ndarjes së Klasës ekuivalente (Equivalence Class PartitioningAnalysis):

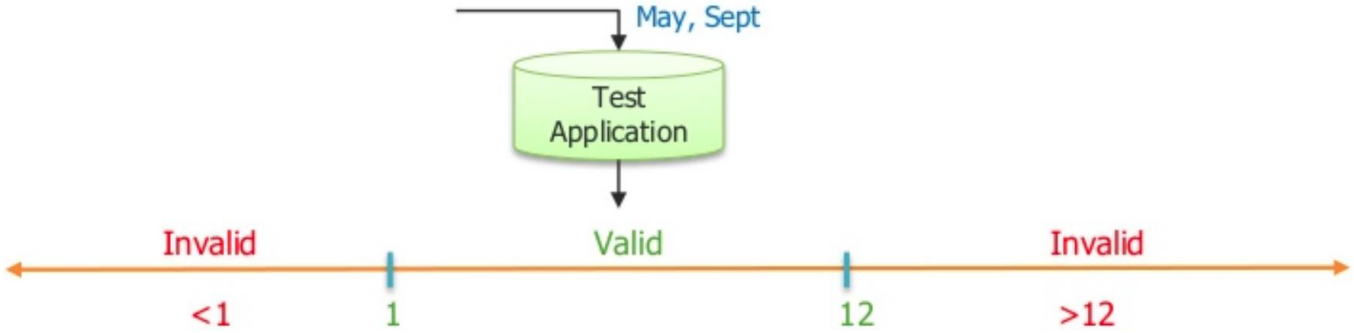
- Në këtë teknikë, ne ndajmë 'Sistemin nën Test' në numrin e klasave të ekuivalencës dhe vetëm i testojm disa/min vlera nga secili klasë.
- Grupi i vlerave i të dhënave që japin një rezultat/dalje të vetme quhet 'ndarje' ose 'klasa', dhe
- Nëse softueri sillet në mënyrë të barabartë me inputet nga ajo klasë atëherë quhet 'Ekuivalencë'. Prandaj, termi Klasa Ekuivalente



Testimi i sipas metodës **kutia e Zezë** (Black-box Testing) (4)

❑Ndarjes së Klasës ekuivalente/Equivalence Class Partitioning(ECP):

- Zakonisht nuk është e mundur që të preformoj testimin e plotë për shkak të kohës ose kufizimeve në buxhetin e projektit.
- Ndarjes së Klasës ekuivalente është **teknikë** shumë efektive e përdorur në testim, kjo form zëvoglon numrin e të dhënave për të **testuar** në aplikacionin

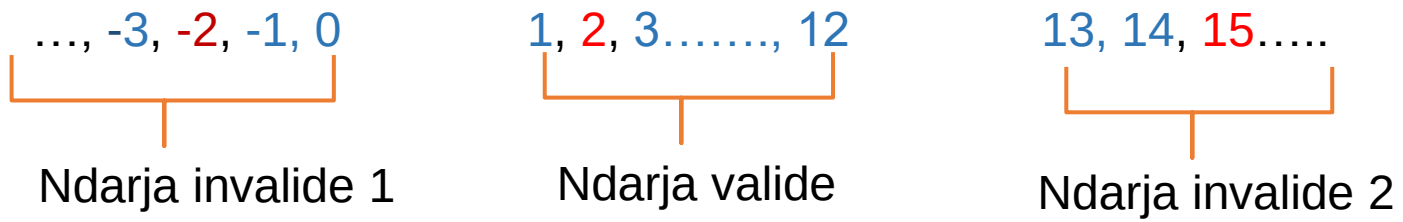


Për të qenë më efikas në kohë dhe buxhet dhe për të mos testuar të gjithë numrat ndërmjet 1 dhe 12, numrat <1 dhe> 12, ne ndajini të dhënat e testimit në 3 grupe, së pari për ndarje të vlefshme (klasa: 1 deri në 12, sekondë për ndarje të pavlefshme (<1) ndarje e pavlefshme (> 12)

Testimi i sipas metodës **kutia e Zezë** (Black-box Testing) (5)

Aplikimi ndarjës së klaseve ekujvalente (ECP):

- Sipas shembullit tonë të dhënat e testimit mund të ndahen në klasa të mëposhtme



- Tani për të testuar aplikacionin testuesi mund të përdorin vetëm një numër nga secilen ndarje
- Pra, ne mund ta testojm aplikacionin me vetëm tre numra (nga një numër i përzgjedhur nga secila ndarje)

Çdo numër i një ndarje të veçantë të **klasës** do të **gjenerojë të njëjtin rezultat**. Kjo është arsyeja pse kjo teknikë quhet **ndarja e klasës së ekuivalencës.**

Testimi i sipas metodës **kutia e Zezë** (Black-box Testing) (6)

❑Vlerave kufitare (Boundary Values):

- Ndarja e Ekuivalencës e Klasave nuk **garanton** testimin e aplikacionit në vlerat kufitare. Ajo ju jep të njëjtën peshë për të dy vlerat në ndarje ekujvalente dhe vlerave e kufirit

…, -3, -2, -1, 0 — Ndarja invalide

1, 2, 3……., 12 — Ndarja valide

13, 14, 15….. — Ndarja invalide

- Analiza e vlerave kufitare është një teknikë e rëndësishme pasi dihet gjerësisht se vlerat në kufijtë shkaktojnë më shumë gabime në **system.**
- Prandaj një testues duhet gjithmonë të kontrollojë kufijtë sikur sistemi të dështojë, ka të ngjarë të dështojë në këto kufij vendimesh.

Testimi i sipas metodës **kutia e Zezë**
(Black-box Testing) (7)

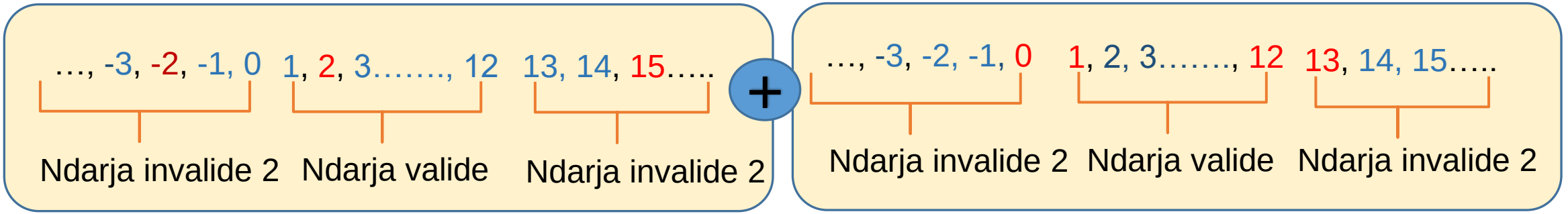
Applying analizen e Vlerave kufitare (Boundary Values BVA):

- Consider the previous example where we partition the test data into three classes
- In boundary values analysis a tester must also check on boundary values in addition to other test data.
- Lets apply BVA

- To apply BVA, we will take the minimum and maximum (boundary) values from the valid partition (1 and 12 in this case) together with first or last value respectively in each of the invalid partitions adjacent to the valid partition (0 and 13 in this case)

Testimi i sipas metodës **kutia e Zezë** (Black-box Testing) (8)

- **Aplikimi ECP dhe BVA së bashku**



- Sipas shembullit ne do të kemi tri teste për ndarjen e ekuivalencës (një nga secila prej tri ndarjeve) dhe katër teste të vlerave kufitare

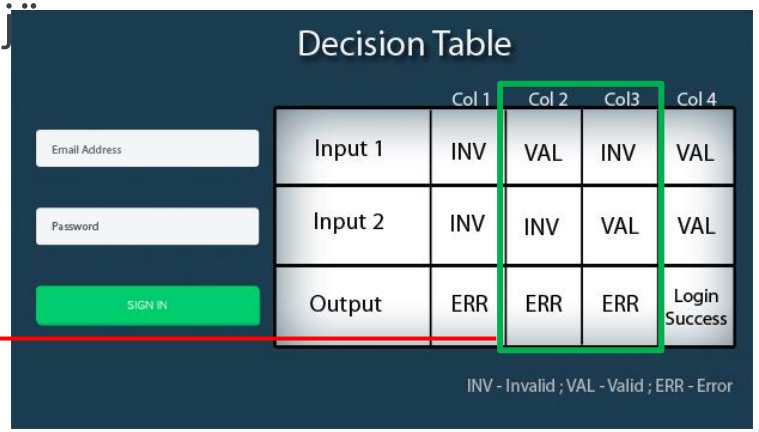
- Për të kryer testet ECP dhe BVA, një testues mund të përdorë një nga kombinimet e mëposhtme të të dhënave të testimit:
  - [-3, 6, 20] (for ECP tests) and [0,1,12,13] (BVA tests]
  - [-4, 4, 14] (for ECP tests) and [0,1,12,13] (BVA tests]
  - [-1, 5, 15] (for ECP tests) and [0,1,12,13] (BVA tests]

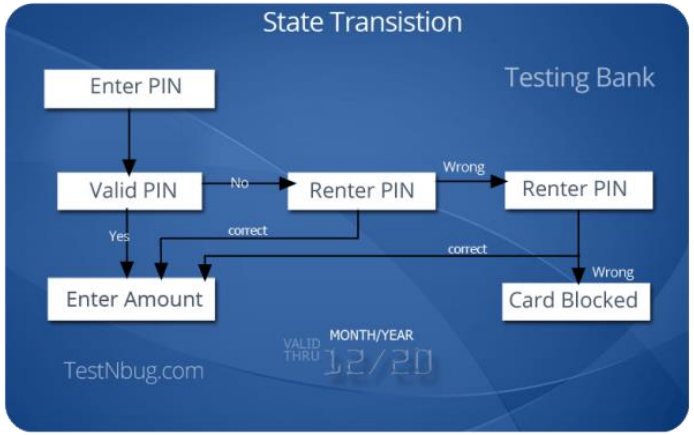
Testimi i sipas metodës **kutia e Zezë**

Testimi i gjendjës/kalimit të Tranzacionit (State Transition Testing) ose Tabelat e vendimeve (Decision tables)

- **Shembull :**
  - Konsideroni automati ATM dhe ju doni të **tërhiqni para** duke përdorur kartën e debitit / kreditit.
  - Nëse përdoruesi **fut fjalëkalim të gabuar-Aplikacioni** do të kërkojë të *futni fjalëkalimin përsëri*.
  - Nëse përdoruesi fut fjalëkalimin e **gabuar për herë të dytë**, aplikacioni do të kërkojë nga përdoruesi që **të fut përsëri**.
  - Herën e **tretë** nëse përdoruesi **fut fjalëkalimin e gabuar** - **Kartela** do të **bllokohet**.
  - Kështu që do të ketë **3 faza të ndryshme të aplikimit** e cila nuk është gjë tjetër veçse tranzicioni i **gjendj**

Kombinimi më **efikas**, duke testuar potencialisht **të gjitha opcionet** për vlera **valide** dhe **invalide**

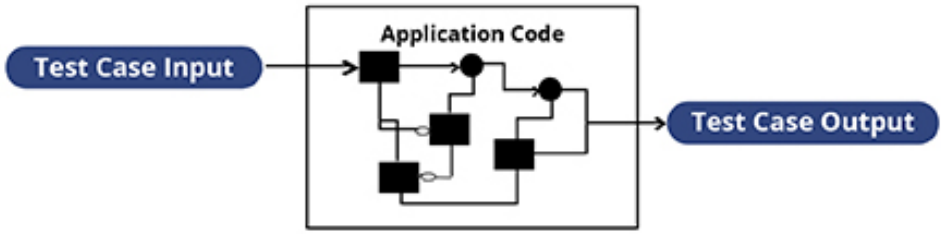


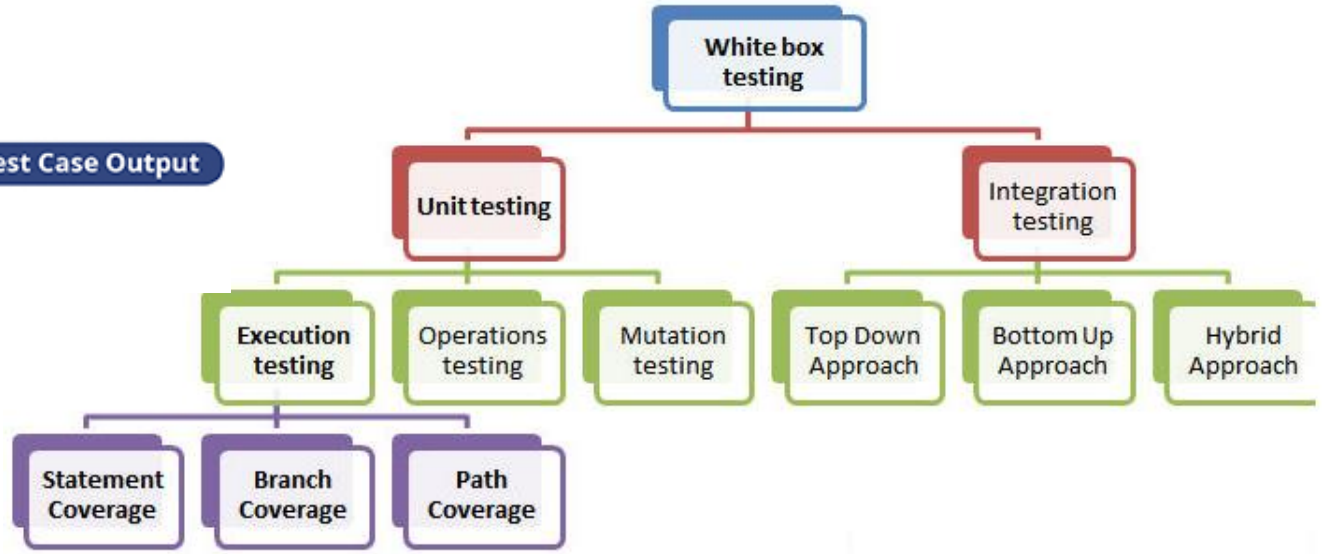


Testimi sipas metodës së **kutis së bardhë**

Testimi sipas metodës së **kutis së bardhë**

- Llojet e testimi sipas ***kutis së bardhë (white box testing method)***
  - Testimi White-box ka të bëjë me teknikat për dizajnimin e testeve; kjo nuk është një nivel i testimit..!!!.
  - Testimi bazuar në analizën e logjikës së brendshme (dizajn, kodi, etj). (Po ashtu, rezultatet e pritshme ende vijnë nga kërkesat.), njohur edhe si testim strukturore.

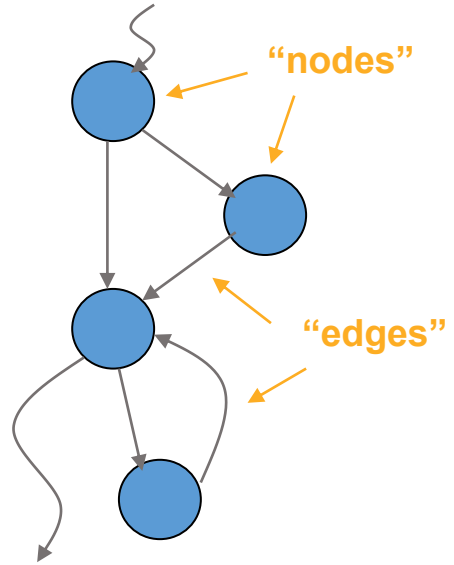




Testimi sipas metodës së **kutis së bardhë**

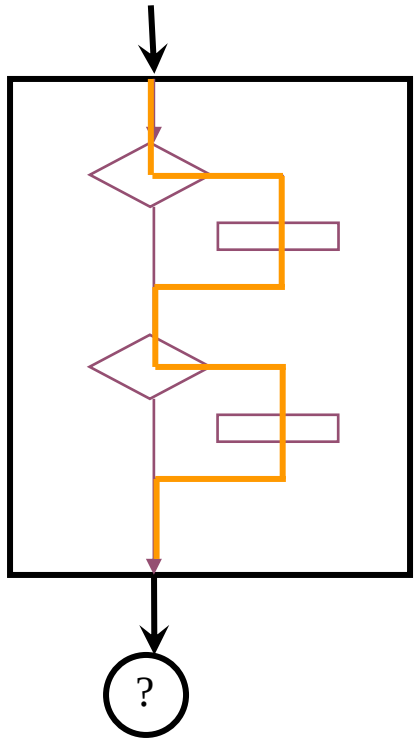
- Trajtimi/mbulim i **deklaratës (të gjitha-nyjet)**-Statement coverage (allnodes)
  - Sigurohuni që çdo rresht i kodit të testohet.
- Trajtimi/mbulim i **degëve (të gjitha-skajet)**-Branch coverage (alledges)
  - Sigurohet që çdo degë (p.sh. e true ose false) është testuar.
- Trajtimi/mbulim i **rruëgve (të gjitha-rrugët)**-Path coverage (allpaths)
  - Sigurohuni që të gjitha shtigjet e mundshme të testohen.

```
input(Y)
        if (Y<=0) then
            Y := -Y
      end_if
   while (Y>0) do
      input(X)
      Y := Y-1
   end_while
```



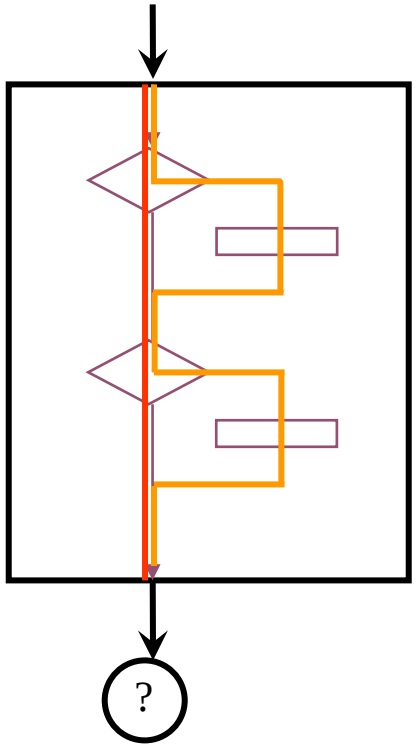
Testimi sipas metodës së **kutis së bardhë**

Statement (deklarata)



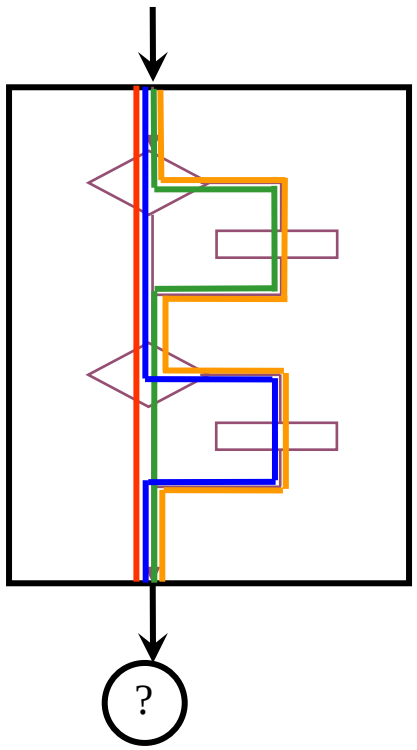
1

Branch (dega)



2

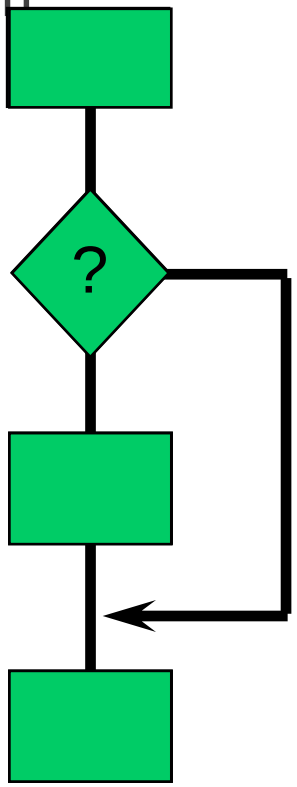
Path coverage (rruga)



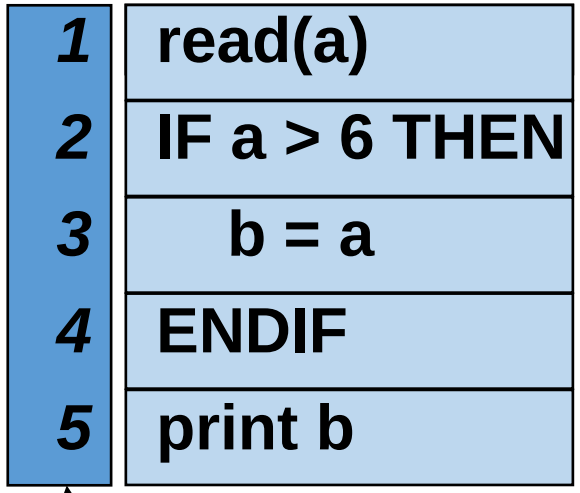
4 Test raste(Test cases)

Kriteret e Përfundimit të Testit dhe Vlera e SC (1)

- Trajtimi/mbulim i **deklaratës (të gjitha-nyjet)**-Statement coverage (allnodes)
- përqindja e deklaratave të ekzekutueshme të ushtruara nga një rethin testuese
- $trajtimi\ i\ deklarates;\ \ sc = \frac{nr\ i\ deklaratave\ te\ testuar}{nr\ total\ i\ deklaratave}$
- shembull:
  - Programi ka 100 deklarata
  - testet ushtrojnë për 87 deklarata
  - D.m.th mbulimi i deklaratave = 87%



Shembull i mbulimit të deklaratave



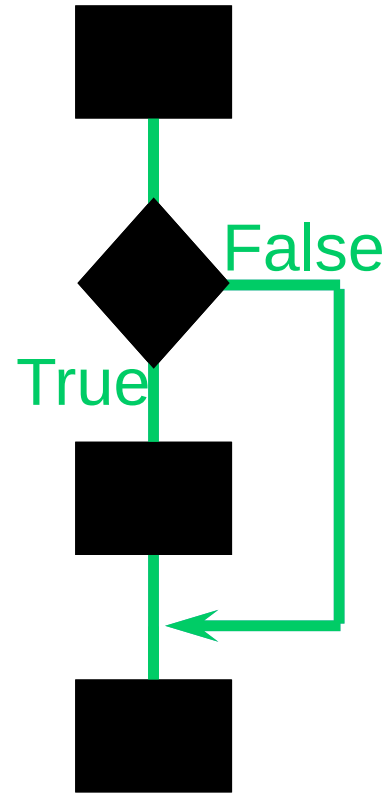
Test rasti	Input	Rezultati i pritur
1	7	7

Numri i deklaratave

Pasi të gjitha 5 deklaratat janë 'mbuluar' nga ky rast testimit, ne kemi arritur 100% mbulim e deklarimit

Kriteret e Përfundimit të Testit dhe Vlera e BC

- Trajtimi/mbulim i **degëve (të gjitha-skajet)**-Branch coverage (alledges)
- Hulumtimet në industri kanë treguar se përmes testimit funksional arrin vetëm 40% deri në 60% të mbulimit të vendimeve/degëve.
- **Vendimi/mbulimi i Degës** është më i **fortë** se **mbulimi i deklaratave** dhe kërkon më **shumë raste testimi** për të **arritur** mbulimin e vendimit 100%.



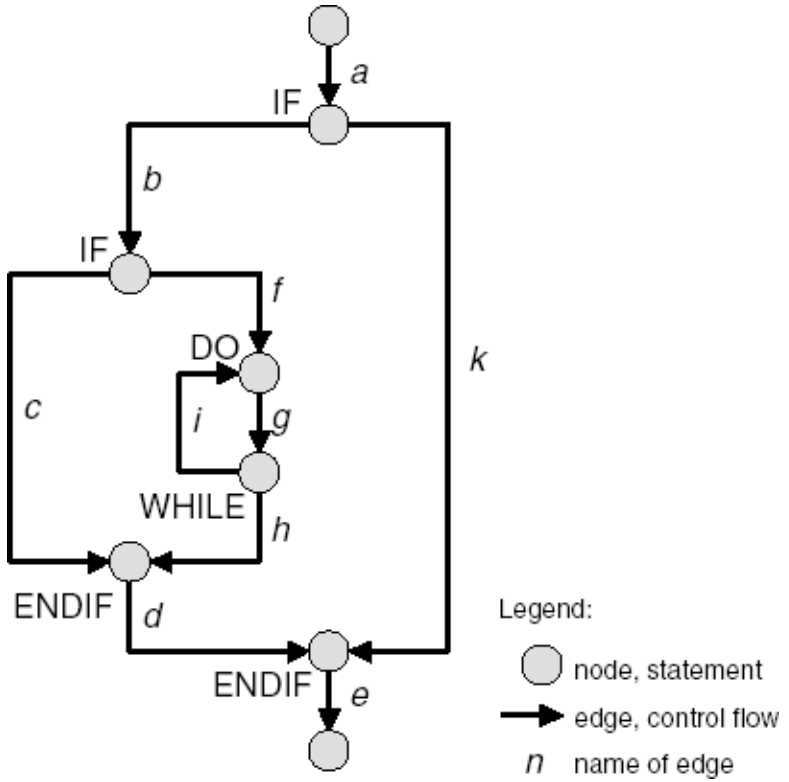
...in e vendimit:

```
READ X
READ Y
IF "X > Y"
PRINT X is greater that Y
ENDIF
```

```
/*To get 100% statement coverage only one test case
  is sufficient for this pseudo-code.
  TEST CASE 1: X=10 Y=5 */
```

Rast i Testimit (Test Cases)

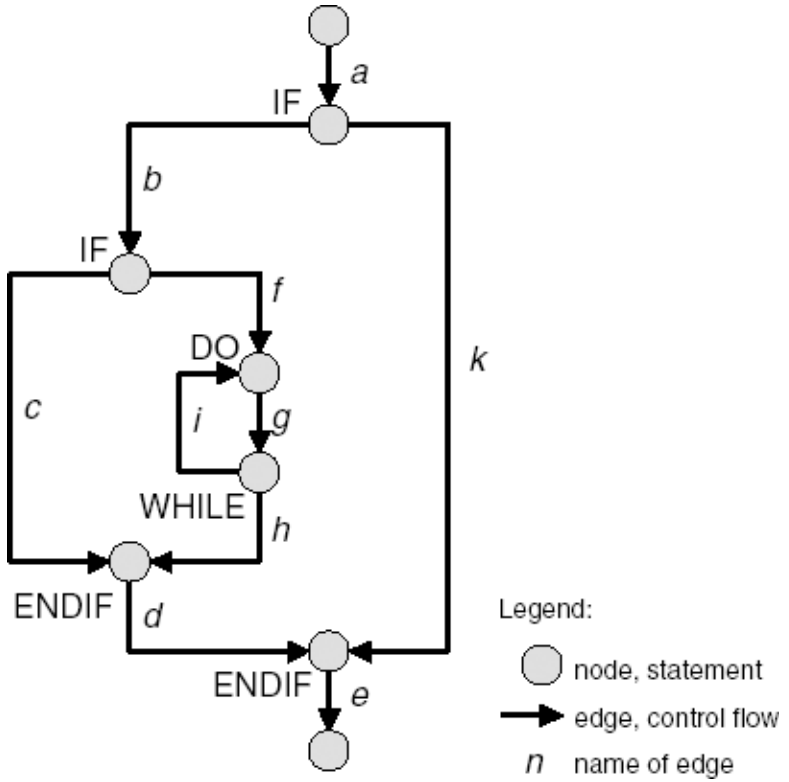
- **Mbulimi i deklaratës** - ekzekutimi i një rasti të testimit, të mbulimi ishte i mjaftueshëm sipas drejtimit në degën
  - `Tc1: a, b, f, g, h, d, e`
- **Deget** `c, i,` the `k` nuk janë ekzekutuar ose mbuluar në rastin e testimit më lartë
- Test raste shtesë janë të nevojshëm:
  - `Tc2: a, b, c, d, e`
  - `Tc3: a, b, f, g, i, g, h, d, e`
  - `Tc4: a, k, e`



Rast i Testimit (Test Cases)...

- **Mbulimi i rrugëve**

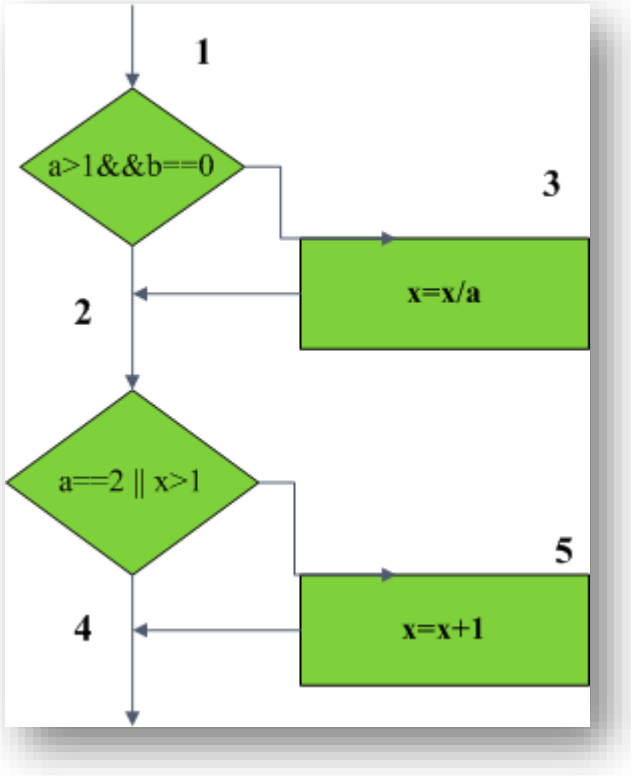
- Të gjitha rrugët e mundshme përmes testimit testohen
- P.sh. Cikli pa përseritje:
  - a, b, f, g, h, d, e
- Cikli(Loop) me kthim të vetëm (i) dhe një përsëritje të vetme :
  - a, b, f, g, i, g, h, d, e
- Zakonisht një cikel (loop) përsëritet më shumë se një herë. Sekuenca të mëtejshme të mundshme të degëve përmes grafikut të programit janë
  - a, b, f, g, i, g, i, g, h, d, e
  - a, b, f, g, i, g, i, g, i, g, h, d, e
  - a, b, f, g, i, g, i, g, i, g, i, g, h, d, etj.
- Kjo tregon se ekziston një numër i pacaktuar i rrugeve në grafikun e rrjedhës së kontrollit



Shembull: Testimi sipas metodës së **kutis së bardhë**

- Trajtimi/mbulimi i deklarates

```
class CovrageTest
{
    public float proc(float a, float b, float x)
    {
        if ((a > 1) && (b == 0))
        {
            x = x / a;
        }
        if ((a == 2)||(x > 1))
        {
            x = x + 1;
        }
        return x;
    }
```

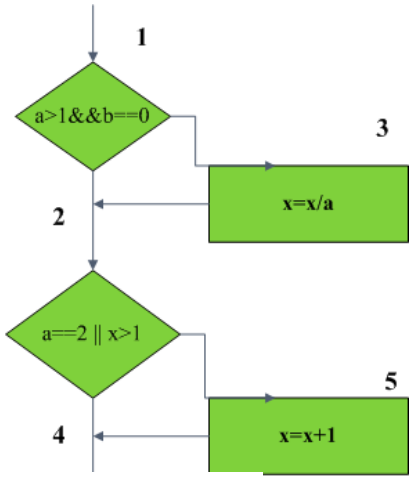


Shembull: Testimi sipas metodës së **kutis së bardhë**

- Rrjedhën e kontrollit të mbulimit:
- **Statement coverage (Mbulimi i deklaratës)**
  - Test Case01 për 1-3-5
    - **P.sh: a = 2, b = 0, x = 3.**
- **Branch coverage (Mbulimi i degës)**
  - Test Case01 për: 1-2-5
    - **P.sh: a = 2, b = 2, x = -1**
  - Test Case02 për: 1-3-4
    - **P.sh: a = 3, b = 0, x = 1**
- **Path coverage (Mbulimi i pathit /rrugë:)**
  - Paths:
    - **1->2->4,**
    - **1->2->5,**
    - **1->3->4,**
    - **1->3->5.**

```
Statment Caverage
Test Case01 (a=2, b=0, x=3)
Statment Coverage x = 2.5
```

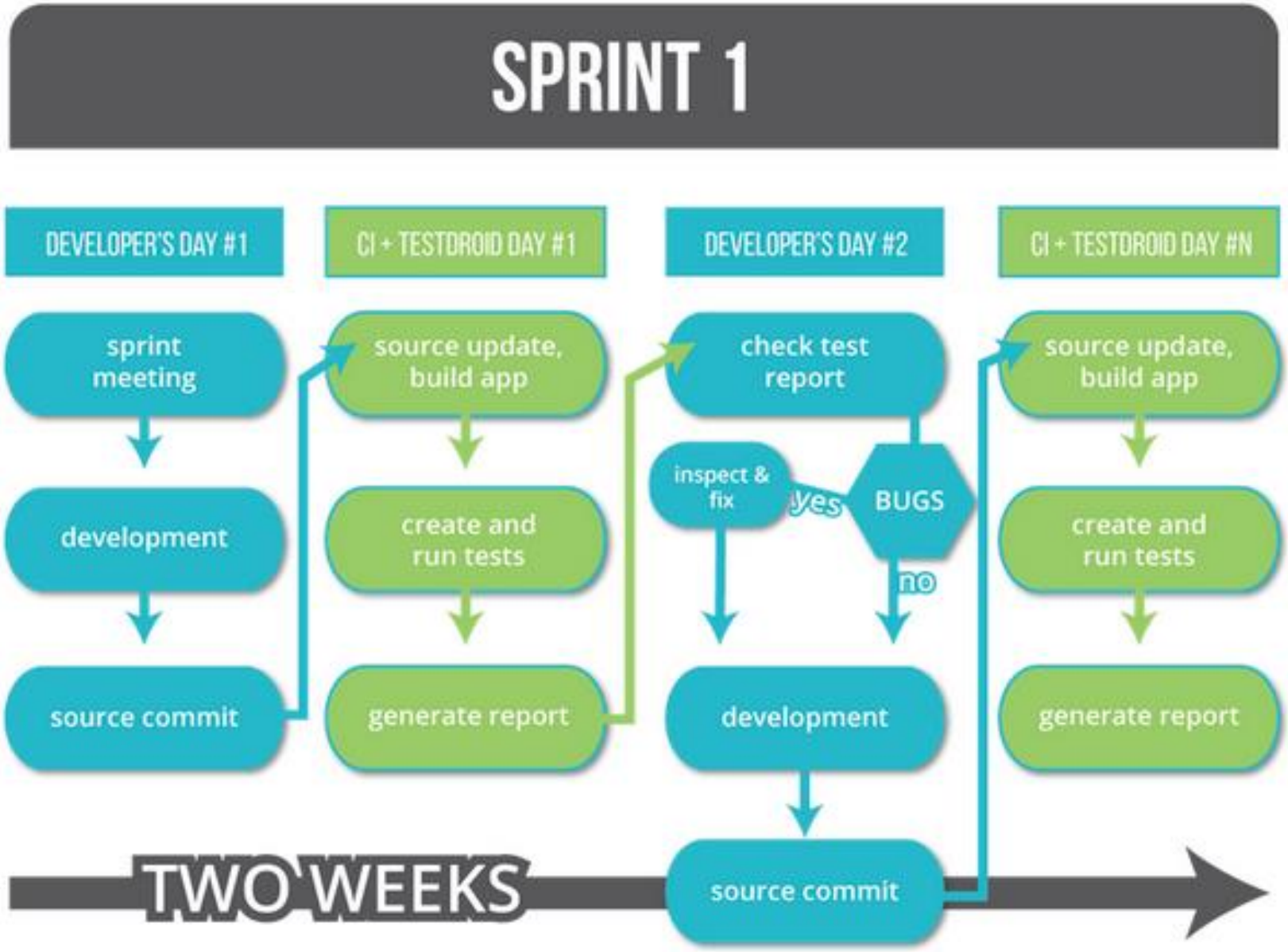
```
Branch Caverage

Test Case01 (a=2, b=2, x=-1)
Branch Coverage  x = 0

Test Case02 (a=3, b=0, x=1)
Branch Coverage  x = 0.3333333
```




```
static void Main(string[] args)
{
    CovrageTest ct = new CovrageTest();
    //Statment Coverage Test Case01
    float sc = ct.proc(2, 0, 3);
    Console.WriteLine("Statment Caverage \nTest Case01 (a=2, b=0, x=3)");
    Console.WriteLine("Statment Coverage x = "+sc+"\n");
    //Branch Coverage Test Case01 1-2-3
    Console.WriteLine("Branch Caverage \n\nTest Case01 (a=2, b=2, x=-1)");
    float bc1 = ct.proc(2, 2, -1);
    Console.WriteLine("Branch Coverage  x = " + bc1);
    //Branch Coverage Test Case01 1-2-3
    Console.WriteLine("\nTest Case02 (a=3, b=0, x=1)");
    float bc2 = ct.proc(3, 0, 1);
    Console.WriteLine("Branch Coverage  x = " + bc2);
    Console.ReadLine();
}
```

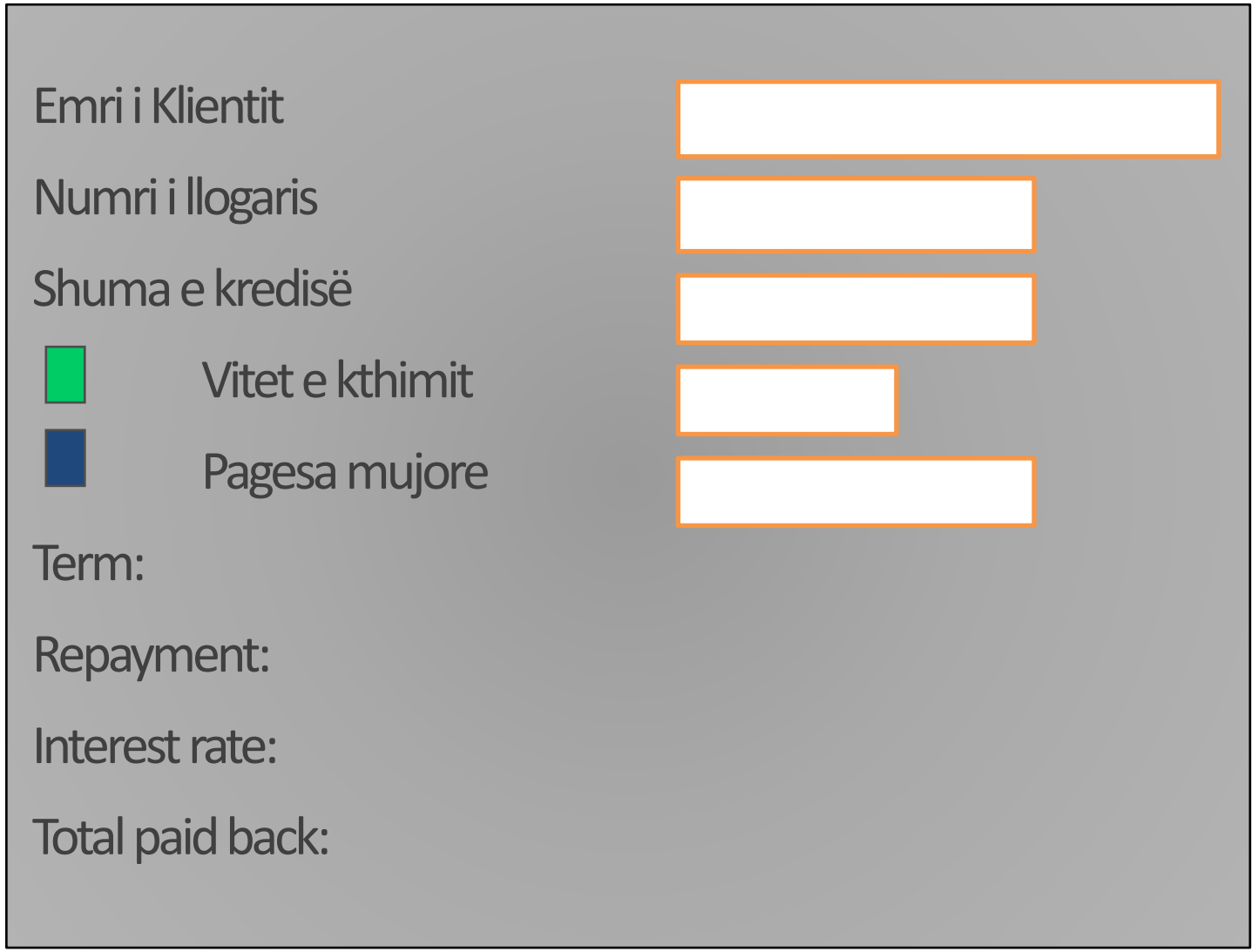
Shembull: Testimit ne modeline XP

THE VALUE OF TESTING IN MOBILE GAME DEVELOPMENT PROCESS



Ushtrime

Shembull: Aplikimi për kredi



2-64 chars.

6 digits, 1st non-zero

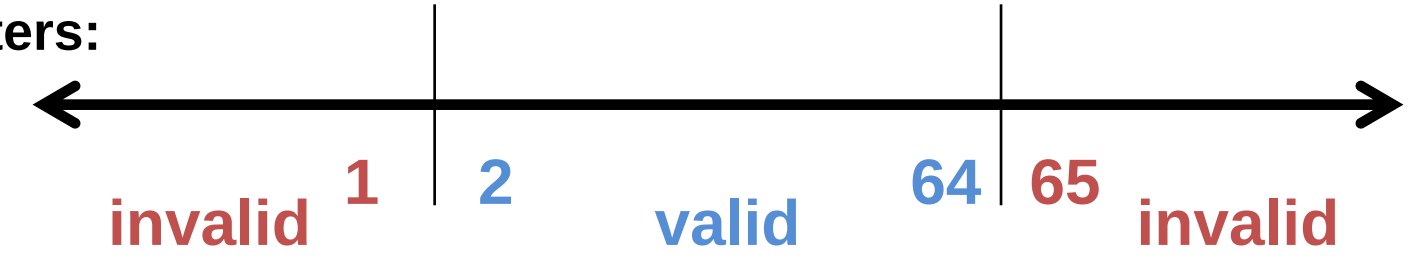
£500 to £9000

1 to 30 years

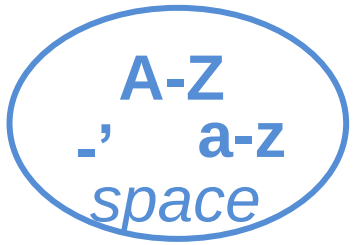
Minimum £10

Emri i Klientit

Number of characters:



Valid characters:



Any other

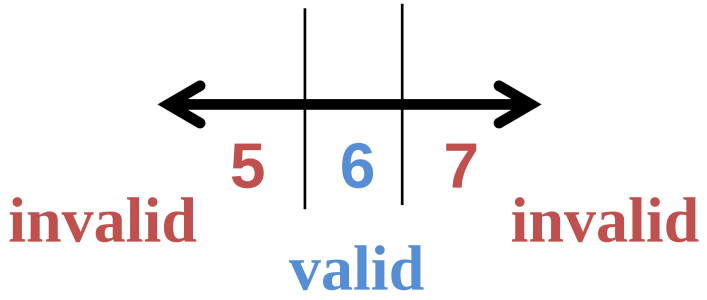
Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Customer name	2 to 64 chars	< 2 chars	2 chars	1 chars
	valid chars	> 64 chars	64 chars	65 chars
		invalid chars		0 chars

Numri i llogaris

first character: valid: non-zero

invalid: zero

number of digits:



Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Account number	6 digits	< 6 digits	100000	5 digits
	1st non-zero	> 6 digits	999999	7 digits
		1st digit = 0		0 digits
		non-digit		

Shuma e kredis

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Loan amount	500 - 9000	< 500	500	499
		>9000	9000	9001
		0		
		non-numeric		
		null		

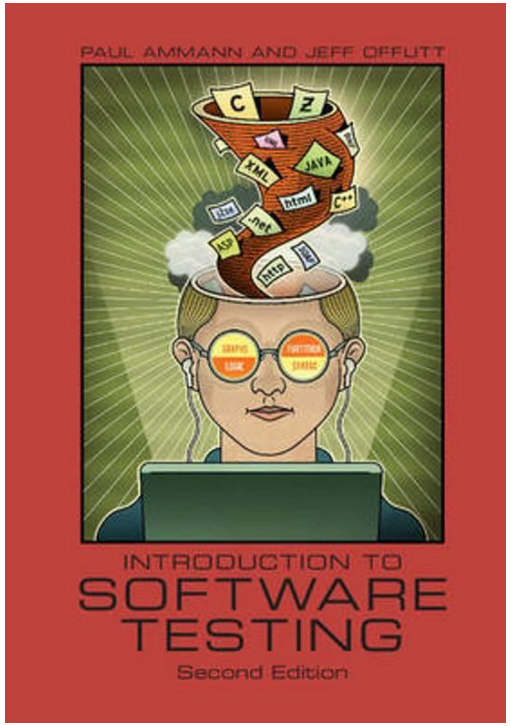
Mostra e vendimeve

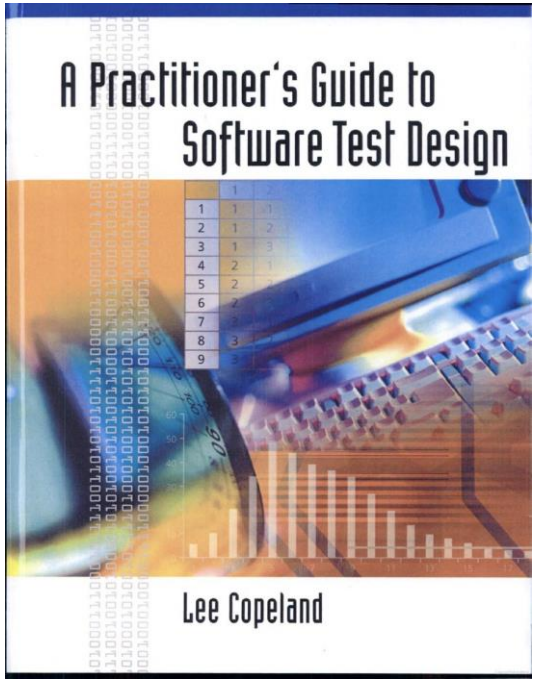
Conditions	Valid Partitions	Tag	Invalid Partitions	Tag	Valid Boundaries	Tag	Invalid Boundaries	Tag
Customer name	2 - 64 chars	V1	< 2 chars	X1	2 chars	B1	1 char	D1
	valid chars	V2	> 64 chars	X2	64 chars	B2	65 chars	D2
			invalid char	X3			0 chars	D3
Account number	6 digits	V3	< 6 digits	X4	100000	B3	5 digits	D4
	1st non-zero	V4	> 6 digits	X5	999999	B4	7 digits	D5
			1st digit = 0	X6			0 digits	D6
			non-digit	X7				
Loan amount	500 - 9000	V5	< 500	X8	500	B5	499	D7
			>9000	X9	9000	B6	9001	D8
			0	X10				
			non-integer	X11				
			null	X12				

Dizajnimi i rasteve të testimit

Test Case	Discription	Expected Outcome	New Tags Covered
1	Name: John Smith Acc No: 123456 Loan: 2500 Term: 3 years	Term: 3 years Repayment: 79.86 Interest rate: 10% Total paid: 2874.96	V1, V2 V3, V4, V5……
2	Name: Dardan Thaqi Acc No: 100007 Loan: 500 Term: 1 years	Term: 1 years Repayment: 44.80 Interest rate: 7.5% Total paid: 537.60	B1, B3, B5,…..

Referenca





Referencat

Kapitulli 8: Software Engineering. 9th ed. By Ian Sommerville

Kapitulli 19: Software Engineering for Students. A Programming Approach. Fourth Edition. by DOUGLAS **BELL**